ירון לייפנברג
אייל לייפנברג




# ASP3

## למפתחי אתרים באינטרנט

הוצאת הוד-עמי
לספרי מחשבים

# ASP 3

## למפתחי אתרים באינטרנט

**הוראות ההתקנה של התקליטור בהקדמה, בפרק 2**

**ובקובץ ONCD שבתקליטור**

עריכה מקצועית: **זהר עמיהוד**

עריכה ועיצוב: **שרה עמיהוד**

עיצוב עטיפה: **שרון רז**

**שמות מסחריים**



**הודעה**



**לשם שטף הקריאה כתוב ספר זה בלשון זכר בלבד. ספר זה מיועד לגברים ונשים כאחד ואין בכוונתנו להפלות או לפגוע בציבור המשתמשים/ות.**

- **טלפון: 09-9564716**
- **פקס: 09-9571582**
- **דואר אלקטרוני: info@hod-ami.co.il**
- **אתר באינטרנט: www.hod-ami.co.il**

# ASP 3

## למפתחי אתרים באינטרנט

ירון לייפנברג
אייל לייפנברג

Designed for Microsoft® Windows NT® Windows®98

הוצאת הוד-עמי
לספרי מחשבים

# ASP 3 for Site Developers

By Y. Lifenberg, E. Lifenberg

Editor: **Z. Amihud**



**הוצאת הוד-עמי**

**לספרי מחשבים בע״מ**

ת.ד. 6108 הרצליה 46160

טלפון : 09-9564716 פקס : 09-9571582

**info@hod-ami.co.il**



הודפס בישראל 2000



**HOD-AMI Ltd.**
P.O.B. 6108, Herzliya
ISRAEL, 2000

ISBN 965-361-232-8 מסת״ב

# תוכן עניינים מקוצר

# תוכן העניינים

# הקדמה

טכנולוגיית ASP תפסה תאוצה אדירה בארץ בשנים האחרונות. רוב מערכות האינטרנט בישראל מבוססות באופן זה או אחר על טכנולוגיית ASP.

עם זאת, עד לכתיבת שורות אלו, לא היה קיים ספר טוב בעברית בנושא, המסכם את הנושאים החשובים ומסביר את ההיגיון מאחורי השימוש בטכנולוגיה והארכיטקטורה אשר מציעים מיקרוסופט באמצעות ASP. פעמים רבות פנו אלינו מפתחים והתלוננו על מחסור בספרות טובה בעברית. לכן, כאשר פנה אלינו המוציא לאור בבקשה לכתוב ספר על טכנולוגיית ASP, חשנו כי הגיע הזמן לספק לציבור מפתחי האתרים בארץ מקור אמין ומקיף שישמש גם כספר לימוד וגם ככלי עבודה שימושי.

לכן, מכיל ספר זה גם תרגילים מתודיים המלווים את המפתח בצעדיו הראשונים, וגם reference מקיף המאפשר גישה מהירה וקלה לדקויות הטכנולוגיה. בספר זה תמצאו גם את הנושאים הבסיסיים ביותר כגון תחביר, חילול דפים דינמיים ושליחת הודעות דואר אלקטרוני, וגם נושאים מתקדמים כגון עבודה מול מסדי נתונים (גם ללא צורך במסד נתונים), יצירת וטיפול קבצים ובניית אובייקטי COM.

אנו מקווים כי תיהנו מהספר ותפיקו ממנו את המירב.

בהצלחה,

**ירון לייפנברג**

**אייל לייפנברג**

# למי מיועד הספר?

ספר זה מיועד ל**מפתחי אתרים** המכירים HTML ו- Dynamic HTML בצורה טובה, וכן שולטים בתכנות ב- JavaScript ברמה בסיסית עד גבוהה.

כמו כן, מיועד ספר זה ל**בוני אתרים** המעוניינים לבצע את קפיצת המדרגה מפיתוח אתרים סטטיים, מבוססי HTML ו- JavaScript בלבד, לאתרים דינמיים המשלבים מידע ממסדי נתונים והיכולים לשלב מערכות שונות תוך שימוש ביכולות כגון הפקת קבצים ושליחת מסרי דואר אלקטרוני באופן אוטומטי.

ספר זה מתאים גם ל**תוכניתני Visual Basic** המעוניינים לרתום את הכוח של סביבת הפיתוח של מיקרוסופט ליישום אפליקציות על גבי האינטרנט או האינטראנט.

למעשה, תוכלו למצוא בספר זה את הבסיס לבניית מערכות מסחר אלקטרוני, בנקאות באינטרנט ויישומי אינטראנט / אקסטראנט.

ללימוד נוסף של הנושאים המוזכרים אנו ממליצים על הספרים הבאים בהוצאת **הוד-עמי**:

**Java 2 למפתחי אתרים באינטרנט**

**JavaScript 1.2 למפתחי אתרים באינטרנט**

**HTML 4 למפתחי אתרים באינטרנט**

**סדנת לימוד Visual Basic 6**

ויש עוד...

את הקטלוג תוכל למצוא בסוף הספר ובתקליטור המצורף.

# על המחברים

**ירון לייפנברג**, סמנכ״ל פיתוח בחברת אינטרנט בינלאומית, מבכירי המומחים בארץ בתחום האינטרנט, האינטראנט והמסחר האלקטרוני. מאחוריו ניסיון רב בפיתוח מערכות אינטרנט מבוזרות מרובות פלטפורמות וכן בייעוץ לגופים מסחריים, צבאיים וממשלתיים בארץ.

**אייל לייפנברג**, מנהל פעילות האינטרנט בחברת מולטימדיה מובילה, הוא מראשוני מפתחי האינטרנט בארץ והראשון שהוסמך על ידי מיקרוסופט ללמד את טכנולוגיית ASP בישראל. מאחוריו שנות הדרכה וייעוץ רבות והובלת פרויקטים המיישמים את קדמת טכנולוגיות האינטרנט.

ספר זה מיישם מתודולוגיה שפיתחו ירון ואייל באלפי שעות הדרכה מול מתחילים ואנשי מקצוע מהשורה הראשונה במרכז ההדרכה של ג׳ון בריס.

# התקליטור המצורף

בתקליטור המצורף לספר זה תוכל למצוא מספר דברים :

- ✓ **קטלוג HTML** - קטלוג ספרי המחשבים האינטראקטיבי של **הוצאת הוד-עמי**. הקטלוג מאפשר קריאת פרקים לדוגמה, תוכן עניינים, מגה-אינדקס ועוד. לשם קריאת הפרקים לדוגמה יש להתקין את תוכנת Adobe Acrobat Reader אשר מצורפת בתקליטור. הקטלוג מומלץ לצפייה באמצעות Internet Explorer 5, המצורפת בתקליטור. התקנת שתי התוכנות קלה וניתנת לביצוע באמצעות קישור ישירות מהקטלוג. הוראות ההתקנה בקובץ ONCD בתקליטור.
- ✓ תוכנת **Personal Web Server**.
- ✓ מספר תוכנות עזר שימושיות.
- ✓ קטעי קוד מקור.

**הערה:** אם מנהל התקן כונן התקליטורים המותקן הוא 16 סיביות (וזה יכול להיות גם אם הכונן חדש) - ייתכן שתראה רק את 8 התווים הראשונים של שם הקובץ (במקרה שהמקור ארוך יותר), או ייתכן שתראה שהתיקיות ריקות. הטוב ביותר הוא להתקין מנהל התקן 32 סיביות, או לקנות כונן תקליטורים חדש ולוודא שמצורף אליו מנהל התקן 32 סיביות.

| **התיקיות הרלוונטיות לספר זה:** | |
|---|---|
| **הוראות התקנה בפרק 2** | **Software\PWS** |
| **הוראות התקנה בהמשך** | **Books\59262** |

**אזהרה:**
השימוש בתקליטור זה הוא על אחריותו הבלעדית של המשתמש. המוצרים המותקנים בתקליטור זה מסופקים באחריות החברות המייצרות אותם. הוצאת **הוד-עמי** אינה אחראית, בכל צורה שהיא, לאופן ולטיב התוכנות המותקנות. הוצאת **הוד-עמי** מפיצה תוכנות אלו כבונוס ללקוחות ההוצאה ואינה מתיימרת לגבות תשלום עבור התוכניות המצורפות ו/או לתמוך בהם. בכל שאלה לגבי תוכנה הנמצאת בתקליטור, יש לפנות למפתחי התוכנה (כל תוכנה בנפרד), כפי שמצוין בקבצי העזרה של התוכנה המדוברת. הקבצים הם גרסאות **שיתופיות** (ShareWare) **וחופשיות** (FreeWare).

**הוראות ההתקנה של התוכנות - בתקליטור בקובץ ONCD או בתיקיה הרלוונטית.**

# Books\59262 - העתקת הקבצים לדיסק

כדי להעתיק את הקבצים לדיסק עקוב אחר ההוראות הבאות. שים לב בפרק 2 להיכן יש להעתיק את הקבצים :

1. לחץ על לחצן **התחל**, **תוכניות**, **סייר Windows**.
2. בכונן הקשיח שלך צור תיקיה חדשה בשם כלשהו.

3. הצג את תוכן תיקיה **Books\59262** אשר בתקליטור.
4. סמן את כל הקבצים וגרור אותם לתיקיה שיצרת.

מעתה, בעת התייחסות בגוף הספר לקבצים/תיקיות שבתקליטור, עבור לתיקיה שיצרת.

מכיון שמקור הקבצים הוא התקליטור, הם מסומנים לקריאה בלבד. רצוי לשנות מאפיין זה בדרך זו:

1. דרך הסייר היכנס לתיקיה בדיסק, שבה נמצאים הקבצים שהעתקת.
2. סמן קובץ מסוים או את כולם על ידי Ctrl+A.
3. הצב את סמן העכבר מעל האזור המסומן ולחץ לחיצה ימנית בעכבר.
4. מתפריט הקיצור בחר **מאפיינים**.
5. בטל את הסימון בתיבה **קריאה בלבד** (דאג שתיבה זו תהיה ריקה).
6. לחץ על **החל**.
7. לחץ על **אישור**.

## תיקיה ראשית SoftWare (רשימה חלקית ועשויה להשתנות)

**הערה**: תוכנות להן יש גירסה מיוחדת עבור Windows 2000 יסומנו בתיקיה ששמה מסתיים ב- 2k (למשל, בתיקיה ICQ נמצא קובץ התקנה לתוכנה זו המתאים לכל גירסאות Windows, ובתיקיה ICQ2k נמצא קובץ התקנה עבור מערכת ההפעלה Windows 2000 בלבד). בדרך כלל לחיצה כפולה על שם הקובץ המפורט ברשימה מפעילה את תוכנית ההתקנה.

| שם תוכנה | תיאור | קובץ הפעלה | מבצע |
|---|---|---|---|
| Adobe Acrobat | תוכנה לצפייה בקבצי pdf. תמיכה בעברית | ARme4ENU.exe | התקנה |
| Babylon | תוכנה לתרגום ולבדיקת איות | B30502h3.exe | התקנה |
| ClnSys | מחיקת קבצי dll שאין צורך בהם | ClnSys16.exe | התקנה |
| FontsPekan | גופנים בעברית | FontsPekan.exe | פריסה |
| ICQ | תוכנה לתקשורת אישית באינטרנט | ICQ99b.exe | התקנה |
| Paint Shop Pro 6 | תוכנה ליצירה, לעיצוב ולעיבוד תמונות | Psp601ev.exe | התקנה |
| WinAmp | תוכנה להשמעת קבצי MP3 (מוסיקה) | WinAmp25e.exe | התקנה |
| WinZip | תוכנית לפריסה ולדחיסה של קבצים | WinZip7sr1.exe | התקנה |

## פרק 1

# הכרת טכנולוגיית ASP

כדי להבין את משמעות טכנולוגיית ASP, יש צורך להבין את פרוטוקול HTTP. פרוטוקול HTTP הוא אחד הפרוטוקולים הפשוטים ביותר ובעיקרון פועל כשרת קבצים. כלומר, ניתן להקביל את פרוטוקול HTTP לשרת המאחסן קבצים ויכול לשלוח קובץ עבור כל בקשה המגיעה מלקוח.

נתאר תהליך פשוט של גלישה ברשת לאתר הבית של YAHOO.

בשלב הראשון, מקליד הגולש את הכתובת http://www.yahoo.com בשדה הכתובת בדפדפן.

מתחת לפני השטח, פונה הדפדפן לשרת ה- HTTP של YAHOO. הפנייה היא בקשת Get, כלומר, הדפדפן מבקש לקבל דבר מה מהשרת. תחביר הפרוטוקול המדויק נראה כך :

Get / HTTP/1.1

**תרשים 1.1:**




בקשת HTTP בנויה מהמילה Get המייצגת את בקשת הלקוח לקבל קובץ, מלוכסן (Slash) ומגרסת הפרוטוקול (http/1.1). היות ובקשה זו מהשרת היא בקשה כללית ולא בקשה לקבל קובץ מסוים, על השרת להחליט איזה קובץ לשלוח ללקוח. חשוב להבין כי שרת הוא מחשב ככל מחשב, ולעיתים אינו חזק או מתוחכם יותר מהמחשב הנמצא בכל בית. האלמנט המבדיל בין מחשב רגיל לבין שרת הוא תוכנה המותקנת על המחשב ומאפשרת לאותו מחשב לקבל בקשות מלקוחות ולשלוח כתשובה קבצים. יש הרבה מאוד תוכנות שרת וחלקן אף מתאימות למחשבי PC בעלי מערכת הפעלה Windows 95/98.

בכל תוכנת שרת, יש אפשרות לקבוע איזה קובץ יישלח כתשובה לבקשת Get כללית. קובץ זה מוגדר ברוב תוכנות השרת כ- Default Document או כ- Index וברוב המקרים הוא קובץ HTML.

בשלב הבא, מחזיר השרת את הקובץ המוגדר כ- Default Document אל הלקוח. הקובץ עובר בפורמט ASCII (American Standard Code for Information Interchange). פורמט ASCII מגדיר ערכי מספרים עבור כל אות, כך שלמעשה נשלחות קבוצות מספרים אשר הלקוח מתרגם חזרה לאותיות ומרכיב קובץ טקסטואלי. קובץ טקסטואלי זה מכיל את ההוראות לדפדפן, כלומר, HTML. עם קבלת הקובץ, ממלא הדפדפן אחר ההוראות ומציג בפני הגולש את האתר.

**תרשים 1.2:**



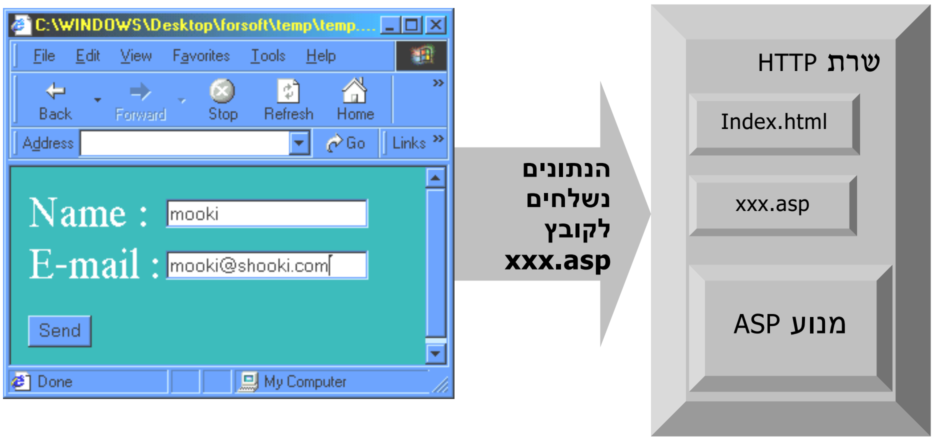

בטכנולוגיה בסיסית זו, יש צורך לכתוב מספר רב של מסמכי HTML. ישנם חסרונות רבים בכתיבת HTML, או כפי שמקובל לכנות טכנולוגיה זו בעגה המקצועית, **אתרים סטטיים**. החסרונות נעים מהצורך לעדכן ידנית נתונים משתנים כגון תאריך, ועד לבעיות חמורות יותר כגון חוסר היכולת להעביר נתונים מהלקוח לשרת ולחולל דפים שתוכניהם משתנים על פי תנאים שונים.

טכנולוגיית ASP מאפשרת לבצע דברים רבים. העיקרון מאחורי ASP הוא לבנות קבצי HTML בזמן הבקשה מהלקוח. כלומר, **אין קבצי HTML מוכנים אלא קבצי HTML נוצרים עבור כל לקוח**. תפישה זו מאפשרת לבנות יישומים נרחבים על סביבת Web, כולל שימוש במסדי נתונים, קבלת נתונים מלקוחות, קישור למערכות שונות, שליחת דואר ועוד.

דוגמה טובה לאתר, המשתמש ביכולות ASP, היא אתר בו המשתמש מזדהה ונרשם באתר. לאחר ההרשמה נשלחת הודעת דואר אלקטרוני אל הלקוח ומברכת אותו על הצטרפותו.

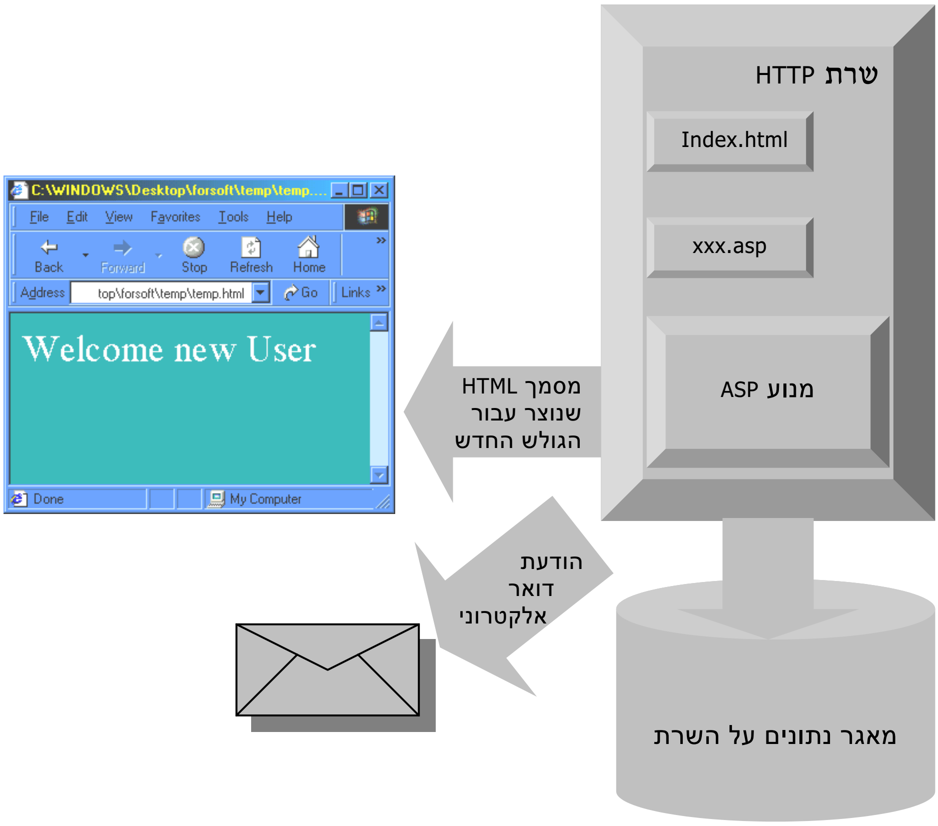
בשלב הראשון שולח הלקוח בקשת Get רגילה ומקבל מסמך HTML רגיל ובו טופס הזדהות/הרשמה. כאשר הלקוח ממלא את הטופס ולוחץ על לחצן השליחה, נשלחים הנתונים אל קובץ בשרת. לקובץ זה סיומת ASP הגורמת למנוע ASP להתעורר ולמלא את ההוראות הרשומות בקובץ. פעולה זו מתבצעת על השרת.

**תרשים 1.3:**




בשלב הבא, בודק מנוע ASP על פי ההוראות הרשומות בקובץ xxx.asp, אם שם המשתמש שהקליד הגולש קיים במאגר נתונים הממוקם על השרת. אם שם הגולש אינו קיים, רושם מנוע ASP את המשתמש החדש במאגר הנתונים, שולח הודעת דואר אלקטרוני לכתובת שהקליד הגולש ולסיום מכין קובץ HTML שמכיל את ההודעה "Welcome New User" ושולח אותו לגולש.

**תרשים 1.4:**




# סיכום

טכנולוגיית ASP מאפשרת בניית יישומים בסביבת Web על ידי מנוע המחולל דפי HTML בזמן ריצה (זמן אמת) מחד, ופקודות גישה למסדי נתונים וביצוע פעולות נוספות מאידך. טכנולוגיה זו תאפשר לנו לייצר דפי HTML עבור כל לקוח באופן המתאים לו ולבקשותיו הספציפיות, ובכך תקצר באופן משמעותי את זמן כתיבת האתר מכיון שקטע קוד אחד יכול להיות אחראי למספר רב של דפים.

## פרק 2

# התקנת מנוע ASP

הפעלה של דפי ASP חייבת להתבצע באמצעות שרת Web (ראה פרק 1, הכרת טכנולוגיית ASP).

שרת WEB של מיקרוסופט הוא **IIS** (Internet Information Server). תוכנת שרת זו ניתנת להורדה חינם מתוך אתר הבית של מיקרוסופט כחלק מ- Option Pack4 לבעלי מערכת הפעלה Windows NT או Windows 2000. יש תוכנות המאפשרות הרצת ASP על שרתי UNIX ו- LINUX (כגון chilli!soft), אך לא נדון בהן בספר זה. מכיון שספר זה עוסק בגירסה 3.0 של ASP הנתמכת במלואה רק בשרת האינטרנט של מיקרוסופט בגירסה 5 (IIS5), ישנם אובייקטים ושיטות אשר לא ייתמכו אם תבחרו לעבוד עם שרתים אחרים כגון Personal Web Server המוסבר בהמשך. שיטות ואובייקטים השייכים לגירסה 3.0 יסומנו בספר זה.

**בסיום ההתקנה יש לאחסן את הקבצים שתצרו בתיקיה Inetpub\wwwroot ולפנות אליהם כ- 127.0.0.1/file.asp. אין לפנות לקבצים אלו דרך open<file של הדפדפן!!!**

**בכל פעם בה תפנו לקבלת קבצים מהשרת הם יילקחו מתוך תיקיה זו.**

**מומלץ לבצע בדיקת תקינות בעזרת תרגיל הבדיקה בהמשך פרק זה.**

לבעלי Windows 95/98 - יש להתקין את תוכנת **Personal Web Server**. את PWS תוכלו למצוא בתקליטור ההתקנה של Windows 98 או בתקליטור המצורף לספר זה, בתיקיה \Software\PWS.

## התקנת Personal Web Server

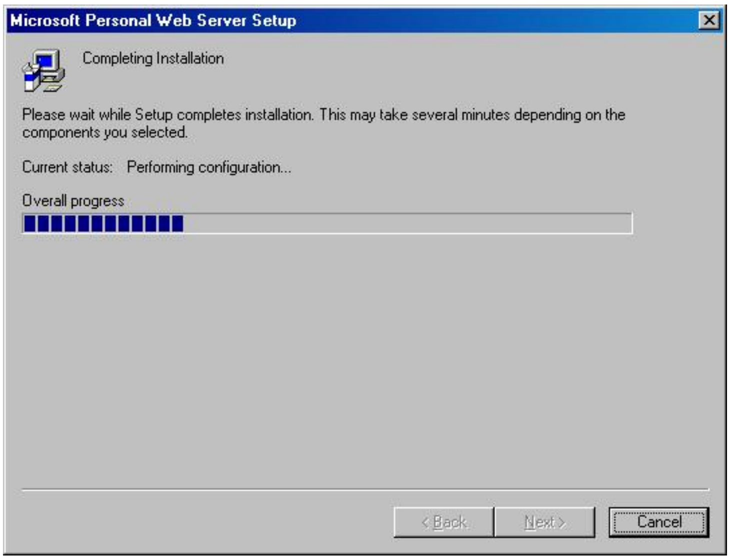
התקנת Personal Web Server במחשב בו מערכת ההפעלה המותקנת היא Windows 95 דורשת עדכון ל- Winsock 2.0. עליכם להתקין את קובץ העדכון W95ws2setup.exe שבתיקיה \Software\Upgrade\ בתקליטור המצורף לספר זה.

לאחר ביצוע העדכון יש לאתחל את המחשב. לאחר שהמחשב "עלה" מחדש:

1. הפעילו את הקובץ Setup.exe שבתיקיה \Software\PWS בתקליטור המצורף לספר, או לחילופין, במחשב בו פועלת מערכת ההפעלה Windows 98, הפעילו את קובץ ההתקנה של Personal Web Server בתיקיה X:\add-ons\pws שבתקליטור ההתקנה המקורי של Windows 98 (החלף את האות X באות המייצגת את כונן התקליטורים שלך).
2. בחלון הפתיחה לחצו על **הבא** (Next).

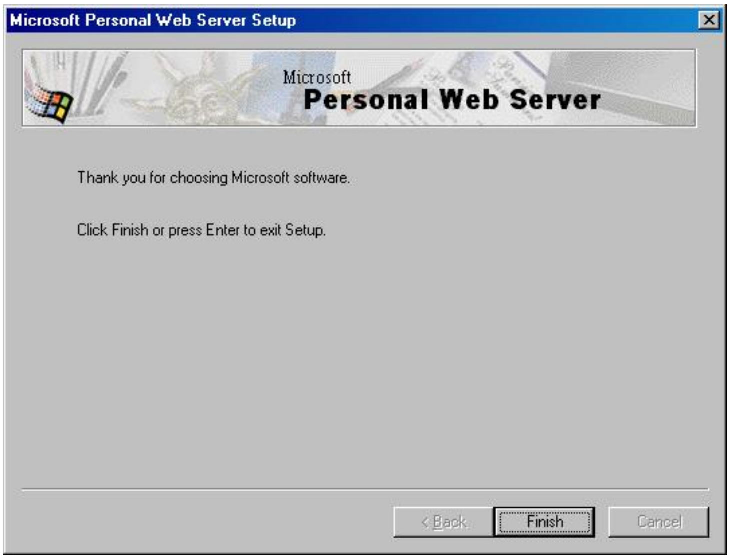



3. בחלון הסכם השימוש בתוכנה (EULA) (מופיע בעת התקנה במערכת ההפעלה Windows 95 ולעיתים מופיע ריק) לחצו על **Accept**.
4. בחלון בחירת סוג ההתקנה לחצו על **Typical** (תוכלו לקרוא את פירוט אפשרויות ההתקנה בחלון).
5. בחלון התקנת תיקיית ברירת המחדל לדף הבית לחצו על **הבא** (Next), או שנו את הנתיב לפי ראות עיניכם (מומלץ להשאיר את ברירת המחדל כמו שהיא. בהמשך הספר ההתייחסות היא לתיקיית ברירת המחדל, C:\Inetpub\wwwroot).




ההתקנה מתבצעת וקבצים מועתקים למחשב.

6. לסיום ההתקנה יש ללחוץ על **סיום** (Finish) ולאתחל את המחשב.



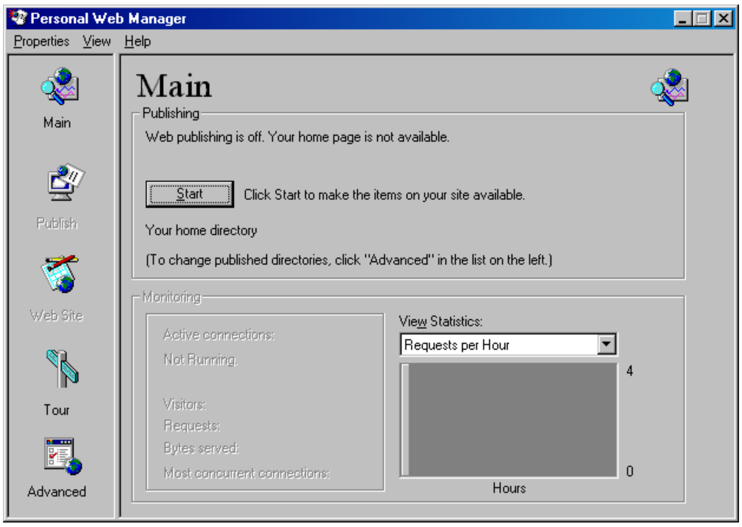

כעת מותקן Personal Web Server במחשבכם.

# הגדרת תצורה של Personal Web Server

בסמוך לשעון בשורת היישומים תוכלו להבחין בסמל הבא: . סמל זה מראה לנו ש-Personal Web Server פעיל.

אם סמל זה אינו מופיע, יש להפעיל את Personal Web Server מתוך תפריט: **התחל**>**תוכניות**>**Personal Web Server**>**Personal Web Manager**.

במסך זה יש ללחוץ על לחצן **Start**.




לאחר הלחיצה יופיע המסך.

מעתה יש לאחסן את הקבצים שתצרו בתיקיה C:\Inetpub\wwwroot (אלא אם שיניתם את תיקיית ברירת המחדל לדף הבית, במהלך ההתקנה).

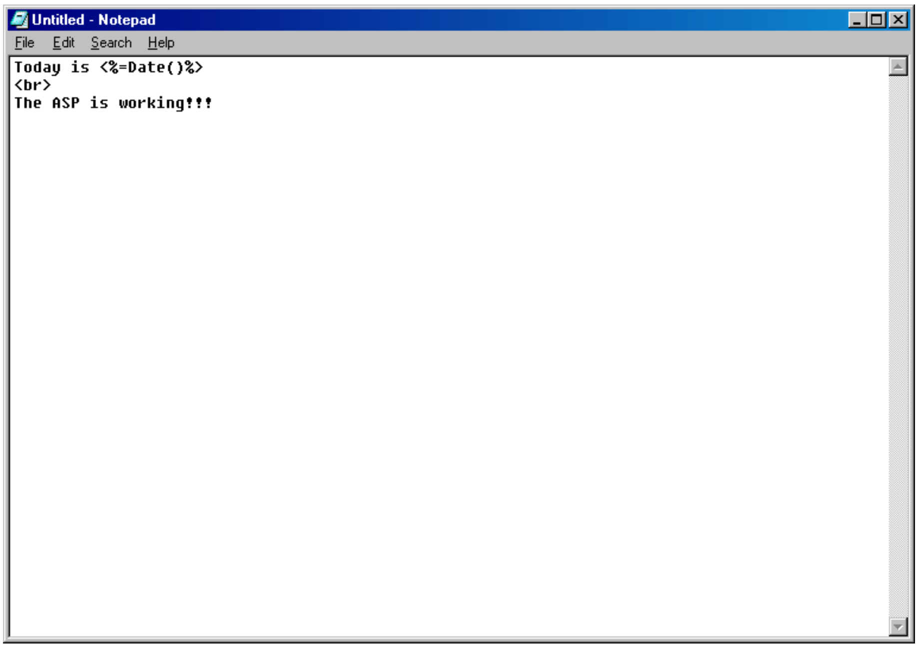
כעת ניגש לבצע בדיקת תקינות.

# תרגיל בדיקת תקינות

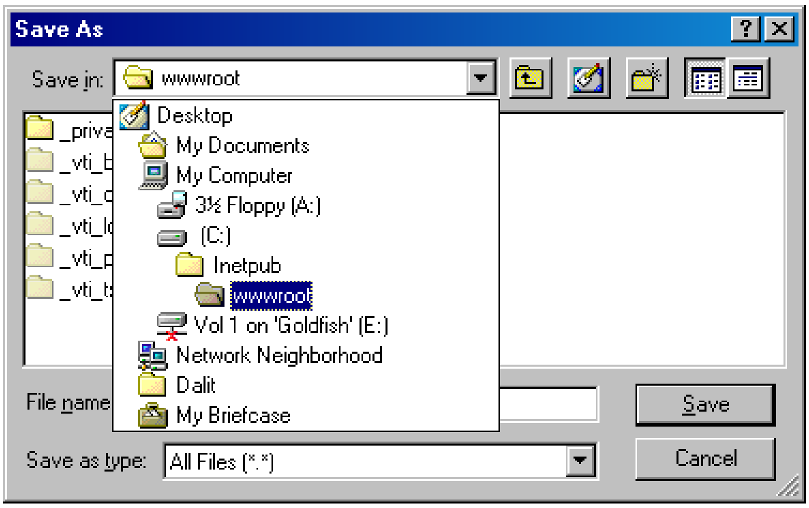
הפעילו את פנקס הרשימות והקלידו את הקוד הבא:

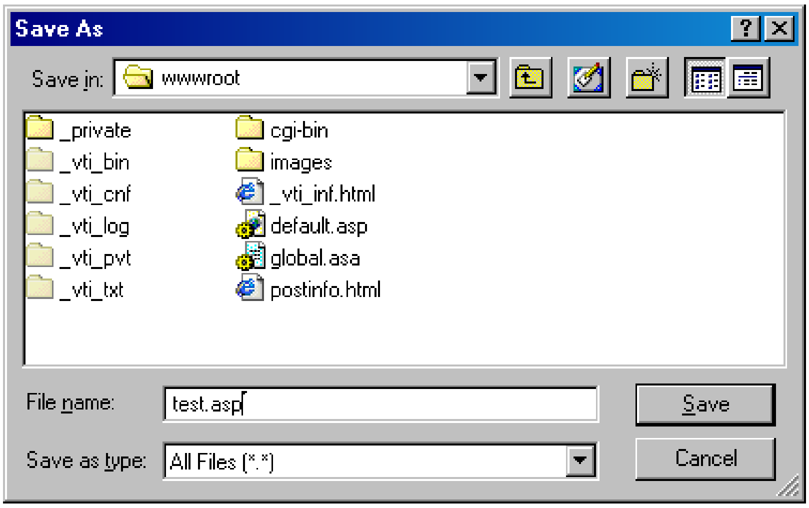
```
Today is <%=Date()%>
<br>
The ASP is working!!!
```

הקוד בפנקס הרשימות ייראה כך :




את המסמך יש לשמור כקובץ **test.asp** בתיקיה C:\Inetpub\wwwroot.



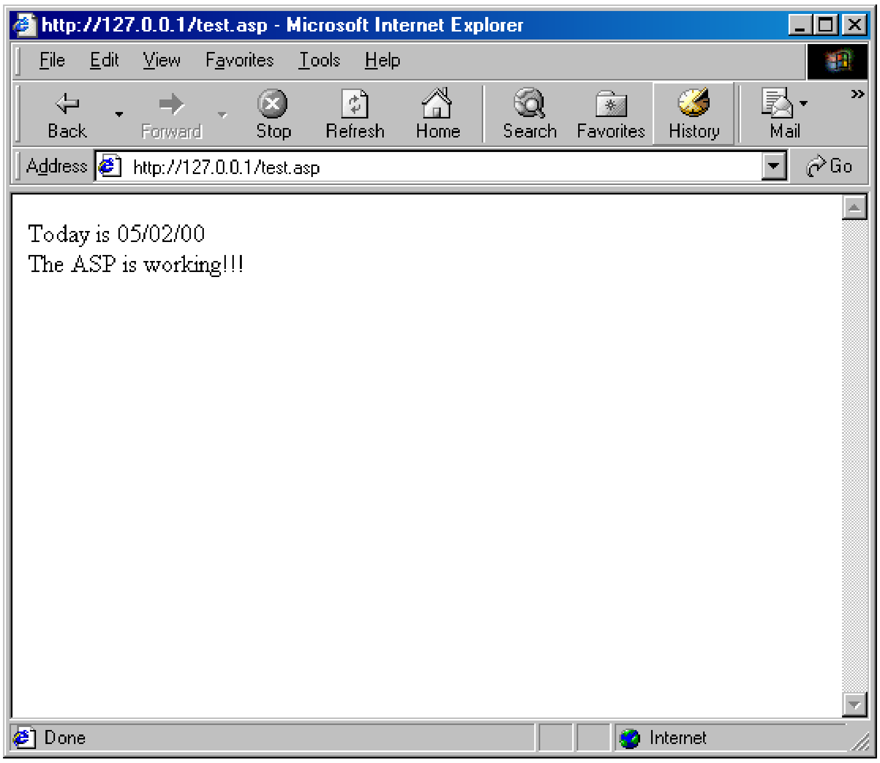



לאחר השמירה של המסמך יש לפתוח דפדפן ולהקליד בשורת הכתובת את הכתובת הבאה ובסיומה להקיש Enter.

127.0.0.1/test.asp

למעשה, שמורה הכתובת 127.0.0.1 לשימוש גלובלי. כלומר, כל מחשב שיקבל את הכתובת 127.0.0.1 יפנה לעצמו (בתהליך שנקרא Loop Back). אם ידועה לכם כתובת IP של השרת, מומלץ להשתמש בה במקום 127.0.0.1.

אם כל השלבים הצליחו, התוצאה שתתקבל תיראה כך :



## פרק 3

# תחביר ושורות קוד ראשונות

בפרק זה נלמד את התחביר הבסיסי אשר יאפשר לנו לבנות את היישומים המתקדמים בהמשך.

הנושאים בפרק זה הם:

- הגדרת משתנים
- בקרת זרימה
- לולאות

בסוף פרק זה נבנה מסמכי ASP בסיסיים.

התחביר אליו נתייחס בספר זה הוא תחביר VBScript. שפת VBScript מתחלקת לשפת צד **לקוח** (שפה שתעבוד עם הדפדפן במסגרת HTML), ולשפת צד **שרת** אשר תעבוד רק ביישומים אשר נבנים על השרת. בניגוד לשפת VBScript בצד הלקוח, אשר נתמכת רק בדפדפן Internet Explorer, שפת VBScript בצד השרת מתבצעת ללא קשר לדפדפן. לכן, נוכל לכתוב בשפה זו מבלי לדאוג לתאימות מול הדפדפנים השונים.

כדאי לדעת כי ניתן לכתוב קוד בשפות נוספות כגון JavaScript, Perl וכו'.

ניתן לומר כי ל- ASP שפה משלו המתבססת על שפת VBScript, משום שיש אובייקטים מיוחדים רק לסביבת ASP. בספר זה לא תתבצע הפרדה זו.

עקרונית, השפה אותה מקבל מנוע ASP כברירת המחדל היא VBScript, אולם, ניתן להכריז על השפה בה משתמשים בקבצי ASP באופן הבא:

```
<%@LANGUAGE=VBScript%>
<%@LANGUAGE=JavaScript%>
```

שיטה נוספת להכרזת השפה היא:

```
<SCRIPT LANGUAGE=VBScript RUNAT=SERVER>
</SCRIPT>

<SCRIPT LANGUAGE=JavaScript RUNAT=SERVER>
</SCRIPT>
```

מכיון שבספר זה נשתמש רק ב- VBScript לקוד בצד השרת, לא נשתמש בהגדרות הנ"ל.

שפת VBScript אינה Sensitive Case, כלומר, אין משמעות לשימוש באותיות גדולות או קטנות.

ייתכן שבדוגמאות בספר נשתמש באותיות גדולות וקטנות לצרכי הבנה אך ניתן לכתוב את הפקודות בכל אופן (למעט קטעי קוד הכתובים ב- JavaScript אשר הינה שפה Case Sensitive).

כשם שדפדפנים מבצעים קטעי קוד התחומים בין סוגריים זוויתיים (< >), יודע מנוע ASP לבצע קוד התחום בין (%>) ל- (<%). לדוגמה:

```
<%
ASP Code
%>
```

כמו שפות תכנות רבות, גם מנוע ASP מתייחס לקוד על פי שורות ולכן אין לשבור שורות, כלומר, כל שורה מתבצעת בפני עצמה. אם יש צורך לכתוב קוד המתפרס על מספר שורות, יש להשתמש בסימן הקו התחתי ( _ ). לדוגמה:

```
<%
ASP Code
Long ASP Code _
Continues here
%>
```

# הצהרת משתנים

משתני VBScript, בדומה למשתני JavaScript, הם משתנים גמישים אשר יכולים לקבל כל סוג ערך מסוג Variant. כלומר, אין צורך להצהיר על סוג המשתנה אלא המשתנה יודע להתאים את עצמו לערך המוזן לתוכו. ליצירת משתנה ניתן להשתמש במילה השמורה **DIM**. לדוגמה:

```
Dim x
x="hello"
Dim y
y=12
```

יש לציין, כי אין חובה להשתמש במילה DIM. כמו כן, מכיון שאין הגדרה של סוג משתנה (מספר, מחרוזת וכדי), המשתנה מתאים את עצמו לסוג הערך שהוזן לתוכו.

אם מתעורר הצורך להסב סוג משתנה, ניתן לבצע הסבה (casting) על ידי שימוש בתחביר:

```
x="41"
y=cint(x)
```

המשתנה y יקבל את ערכו של x המוסב ממחרוזת למספר. ניתן להסב משתנים לסוגים רבים כגון:

| | | טווח |
|---|---|---|
| **cint** | החלף לסוג מספרי (integer) | 32768- עד 32767 |
| **cstr** | החלף לסוג מחרוזת (string) | 0 עד 2 מיליארד תווים בקירוב |
| **csng** | החלף לסוג בודד (single) | $-3.402823e^{38}$ עד $1.401298e^{-45}$ / $3.402823e^{38}$ עד $1.401898e^{-45}$ |
| **cdbl** | החלף לסוג כפול (double) | $-1.79769313486232e^{308}$ עד $4.94065645841247e^{-324}$ / $1.79769313486232e^{308}$ עד $4.94065645841247e^{-324}$ |
| **clng** | החלף לסוג ארוך (long) | 2,147,483,648- עד 2,147,483,647 |
| **cbool** | החלף לסוג בוליאני (boolean) | true או false |
| **cbyte** | החלף לסוג בית (byte) | 0 עד 255 |
| **ccur** | החלף לסוג מטבע (currency) | 922,337,203,685,477.5808- עד 922,337,203,685,477.5807 |
| **cdate** | החלף לסוג תאריך (date) | 1.1.100 עד 31.1.9999 כולל |

כמו כן, ניתן להפוך ייצוג עשרוני (decimal) לייצוג בבסיס 16 (hexadecimal) על ידי שימוש בתחביר:

```
x=10
y=hex(x)
```

המשתנה y יקבל את הערך A שהוא הייצוג ההקסדצימלי ל- 10.

הסיכוי להשתמש בכל סוגי המשתנים הוא נמוך ביישומי אינטרנט בסיסיים, אך כדאי להכיר את האפשרויות הקיימות.

ASP מחולל דפי HTML. ניתן לחולל קוד דינמי (כגון משתנים) וכן ניתן להשתמש בקוד סטטי (כגון שורות טקסט או פקודות HTML. שורות אשר אינן תחומות בין סימני הפתיחה (<%) והסגירה (%>) של ASP, יישלחו אל הלקוח כמות שהן. כלומר, הקובץ:

```
<%
dim x
x="Mooki"
%>
<font color=red>Welcome
```

יחולל וישלח אל הלקוח את המסמך:

```
<font color=red>Welcome
```

אחת הדרכים לשלב קוד דינמי בשורות הסטטיות היא להשתמש בסימן השווה (=). לדוגמה, אם נרצה לשלוח קוד המשלב את שורת HTML הסטטית - <font color=red>Welcome עם ערך המשתנה המוגדר במסגרת קוד ה- ASP, נכתוב את הקוד בצורה הבאה:

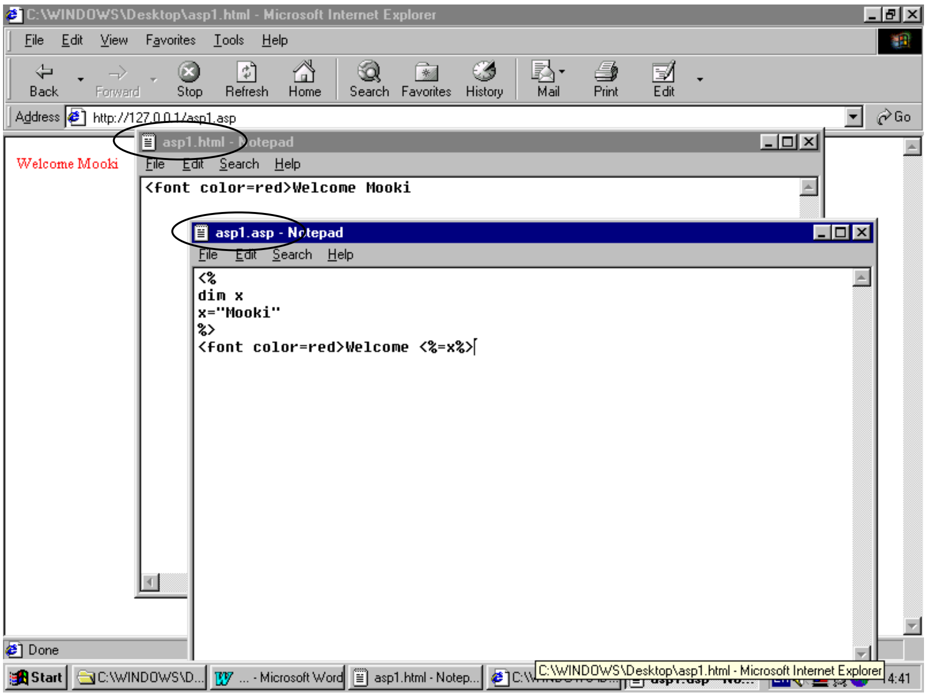
asp1.asp

```
<%
dim x
x="Mooki"
%>
<font color=red>Welcome <%=x%>
```

מסמך ASP זה יחולל וישלח ללקוח את מסמך HTML הבא:

```
<font color=red>Welcome Mooki
```

להלן מסך הדפדפן, הקוד הנראה ב- View Source וקוד ASP:




כלומר, מנוע ASP שלח ללקוח את שורות הקוד הסטטיות (<font color=red>Welcome) ובמקום הקוד הדינמי (<%=x%>), מיקם את ערכו של x, כלומר, Mooki.

לטובת הדוגמאות הבאות, כאשר נרצה לשלב ערך משתנה במסמך HTML הנשלח ללקוח, נפתח את הסימן (<%), נוסיף את סימן השווה (=), נרשום את המשתנה ונסגור עם הסימן (%>).

הדוגמאות הבאות יכילו את מסך הדפדפן כשמעליו מסמך HTML הנשלח ללקוח (אותו ניתן לראות ב- View Source), ולבסוף את קובץ ASP השמור בשרת. דרך הדגמה זו מראה בצורה ברורה כיצד קוד ASP מתבצע על השרת ומכין מסמך HTML המתבצע על דפדפן הלקוח.

# בקרת זרימה ולולאות

כבכל שפת תכנות, גם ב- ASP יש טיפול בבקרת זרימה ובלולאות. מכיון שאנו מניחים שקוראי ספר זה מיומנים בכתיבת JavaScript, אין צורך להסביר את משמעות המושגים, אלא להתמקד בתחביר.

## תחביר IF

ב-VBScript ניתן להשתמש בבדיקת IF באופן הבא:

```
IF Condition THEN
    ASP Statement
END IF
```

להבדיל מ-JavaScript, התנאי אינו תחום בסוגריים ויש להוסיף את המילה THEN באותה שורה.

כמו כן, הפקודות המתבצעות אם התנאי מתקיים, אינן תחומות על ידי סוגריים, אלא מתבצעות עד הפקודה END IF.

ניתן כמובן להשתמש גם במילה ELSE באופן הבא:

```
IF x=2 THEN
  Y=Y+1
  Z=Z-8
ELSE
  Y=Y-1
  Z=Z*9
END IF
```

## Select Case

בדיקה נוספת (מקבילה לבדיקת Switch ב-JavaScript) נקראת CASE, והתחביר לביצוע הבדיקה הוא:

```
SELECT CASE X
CASE 2
  Y=Y+2
CASE 3,4,10
  Y=Y+3
END SELECT
```

המילים SELECT CASE X מגדירות כי הערך הנבחן הוא הערך של המשתנה X. כל שורה הנפתחת במילה CASE מגדירה מקרה בו למשתנה ערך מסוים. לדוגמה, בשורה השנייה מוגדר כי במקרה ולמשתנה X יש ערך 2, יש להוסיף 2 למשתנה Y. בשורה הרביעית,

מוגדר כי במקרה ולמשתנה X ערך מטווח הערכים המופרדים בפסיקים, יש להוסיף 3 למשתנה Y.

להלן דוגמה :

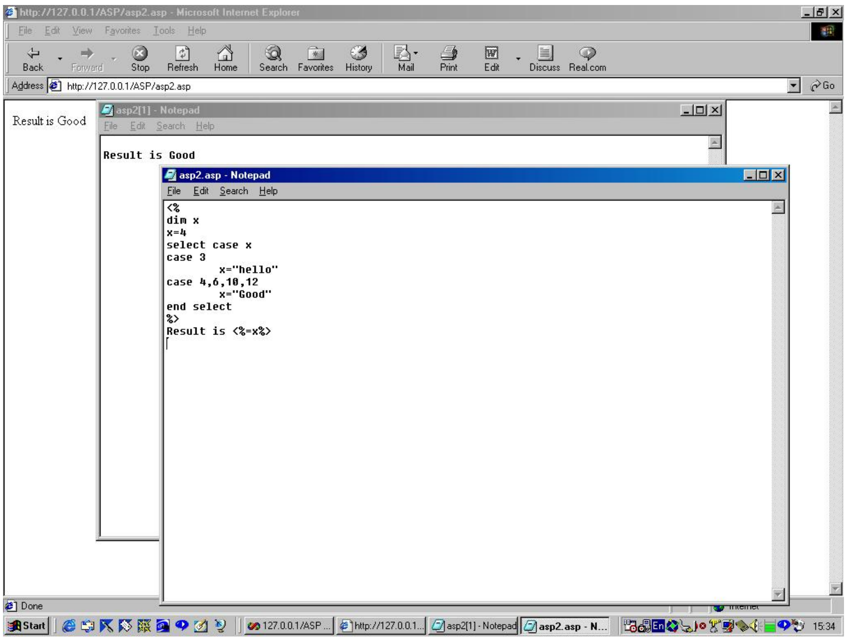
asp2.asp

```
<%
dim x
x=4
select case x
case 3
  x="hello"
case 4,6,10,12
  x="Good"
end select
%>
Result is <%=x%>
```

מסמך HTML שיישלח ללקוח הוא :

```
Result is Good
```

להלן המסך :



# לולאת FOR

התחביר לשימוש בלולאת FOR, גם הוא שונה מלולאת FOR של JavaScript. בלולאת FOR של VBScript יש להגדיר ערך התחלתי של משתנה, ערך סופי, וניתן להגדיר את השינוי בערך המשתנה באופן הבא :

```
FOR I=1 TO 100 STEP 5
  X=X+1
NEXT
```

התחביר כולל את המילה FOR, הערך ההתחלתי של המשתנה המונה את ריצות הלולאה, המילה TO והערך הסופי אליו מגיע המשתנה ומפסיק את ריצת הלולאה. המילה STEP מגדירה כי ערכו של המשתנה יעלה ב- 5 בכל ריצה של הלולאה. אם לא נשתמש במילה STEP, ברירת המחדל היא 1, כלומר, ערכו של המשתנה יעלה ב- 1 בכל ריצה של הלולאה. יש לציין כי בספירה שלילית (משתנה המתחיל עם ערך גבוה מערך הסיום), חובה להשתמש במילה STEP להגדרת שינוי הערך.

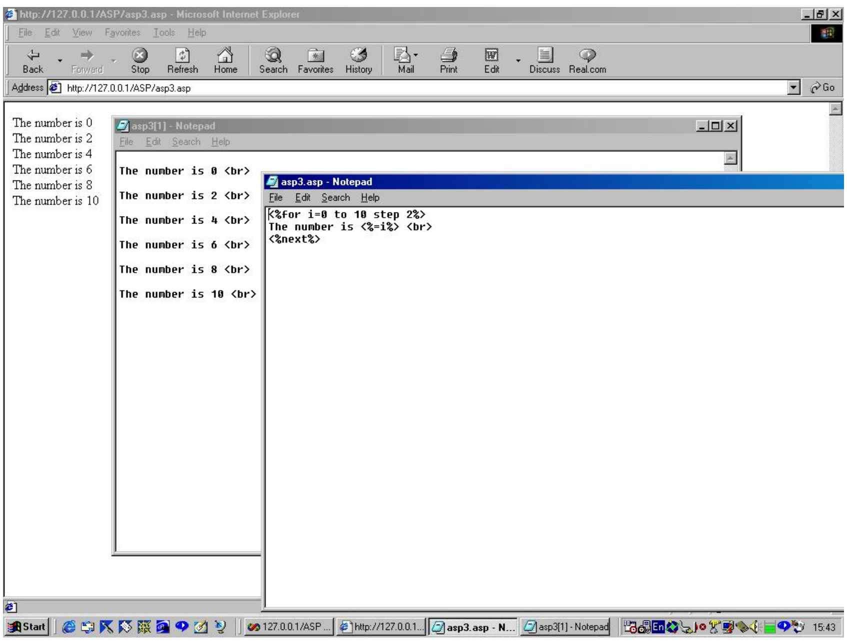
הגדרת סיום הלולאה היא המילה NEXT.

כדי לראות תוצאה של ריצת לולאה, ניתן ליצור את קובץ ASP הבא :

asp3.asp

```
<%for i=0 to 10 step 2%>
The number is <%=i%> <br>
<%next%>
```

להלן המסך :

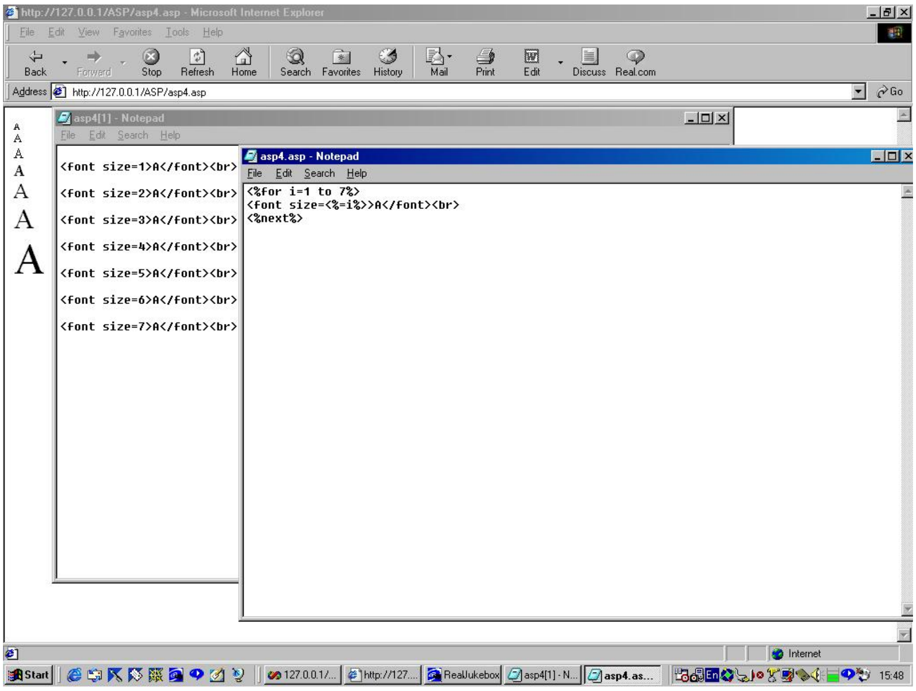


דוגמה נוספת היא :

asp4.asp

```
<%for i=1 to 7%>
<font size=<%=i%>>A</font><br>
<%next%>
```

להלן המסך :




ניתן לעצור לולאת FOR במקרים מסוימים על ידי שימוש בביטוי Exit For. לדוגמה :

```
for I=1 to 10
  if I=7 then
    exit for
  end if
next
```

## לולאת DO UNTIL

לולאה זו מאפשרת לבצע ריצות לולאה עד להתקיימות תנאי מסוים. לדוגמה :

```
DO UNTIL X=2
X=X-1
LOOP
```

המילה התוחמת את גבולות הלולאה היא LOOP.

## לולאת DO WHILE

לולאה זו תתבצע כל עוד מתקיים תנאי מוגדר, כגון:

```
DO WHILE X<2
X=X+1
LOOP
```

גם בלולאה זו, המילה התוחמת את גבולות הלולאה היא LOOP.

שיטה נוספת לכתיבת לולאה זו היא באופן הבא:

```
WHILE X<2
X=X+1
WEND
```

בשיטה זו, המילה התוחמת את תכולת הלולאה היא WEND.

## הערה (remark)

בכל שפת תכנות ניתן לתעד את הקוד על ידי הערות. כדי לכתוב במסמך שורת הערה שלא תתבצע על ידי מנוע ASP, יש להשתמש בסימן התג (') באופן הבא:

```
'this is a remark
```

## אופרטורים

| | |
|---|---|
| And | וגם |
| Mod | מודולוס |
| Not | לא |
| Or | או |
| = | שווה |
| < | קטן מ- |
| > | גדול מ- |
| <= | קטן שווה |
| >= | גדול שווה |
| <> | שונה |

# מערכים

הכרזה על מערך חד-מימדי נעשית על ידי שימוש בפונקציית Array:

```
MyArray = Array ("yossi","Benny","meshulam","Dalit")
```

פנייה לאיבר במערך נעשית על ידי ציון מיקומו בתוך סוגריים מעוגלים (האיבר הראשון נמצא במיקום 0):

```
MyArray(3)="Mendelbaoum"
```

ניתן להגדיר את גודל המערך ללא הזנה ראשונית של ערכים, על ידי שימוש בהכרזת Dim וציון מספר האיברים הייעודי:

```
Dim MyArray(6)
```

כדי ליצור מערכים רב-מימדיים יש להשתמש בהכרזת Dim, ציון מספר המימדים וגודלם. כך לדוגמה, הכרזה על מערך דו-מימדי בעל 3 איברים במימד הראשון ו- 2 איברים בשני תיראה כך:

```
Dim MyArray(3,2)
```

בשימוש במערכים יש חובה לציין את גודלם, אולם לעיתים גודלו של המערך יכול להשתנות.

במקרים בהם נרצה להגדיל או להקטין את המערך יש להשתמש בהכרזת ReDim, המאפשרת לנו להקצות בצורה דינמית גודל חדש למערך. כך לדוגמה, מערך שגודלו הראשוני היה של 5 איברים יגדל בעזרת ReDim ל- 10 איברים:

```
Dim MyArray(5)
ReDim MyArray(10)
```

חשוב לציין, כי שינוי גודל המערך ימחק את תכולתו הקודמת, למעט אם נשתמש בהכרזת Preserve המורה לשמור את תכולת המערך:

```
Dim MyArray(10)
ReDim Preserve MyArray(20)
```

# פרק 4

# אובייקט Request – התחלת הקשר בין השרת ללקוח

בסוף פרק זה נדע לקבל מידע מהלקוח (דרך טופס HTML), ולהחזיר תשובה דינמית.

ל-ASP מספר אובייקטים המאפשרים לבנות יישומים אינטראקטיביים בין השרת ללקוח. בהמשך נבין את משמעות האובייקטים השונים ואף נשתמש בהם. האובייקט הראשון, ואחד המשמעותיים ביותר, הוא **אובייקט Request**. אובייקט זה מאפשר לקבל מידע מדפדפן הלקוח החל בנתונים המוזנים לתוך שדות בטופס וכלה בעוגיות ונתונים נוספים. כדי להשתמש באובייקט זה, יש להבין את פעולת טפסי HTML.

## הבנת טפסי HTML

טופס HTML מורכב ממספר אובייקטים. אובייקט מסוג אחד הוא שדה הקלט (text, button, checkbox, radio...) המאפשר קליטת נתונים מהמשתמש. האובייקט החשוב הבא הוא אובייקט ההגשה (submit) המיוצג כלחצן, אשר עם לחיצתו נשלחים הנתונים שנאספו בטופס.

מכיון שלחיצה על לחצן submit שולחת את הנתונים, יש להגדיר בצורה מדויקת להיכן נשלחים הנתונים. הגדרה זו מתבצעת באובייקט פתיחת הטופס (<form>) על ידי התכונה **Action**. תכונה זו מגדירה לאיזה יישום ועל איזה שרת לשלוח את נתוני הטופס והגדרתה מתבצעת באופן הבא:

```
<form action=http://www.yogli.com/search_application>
```

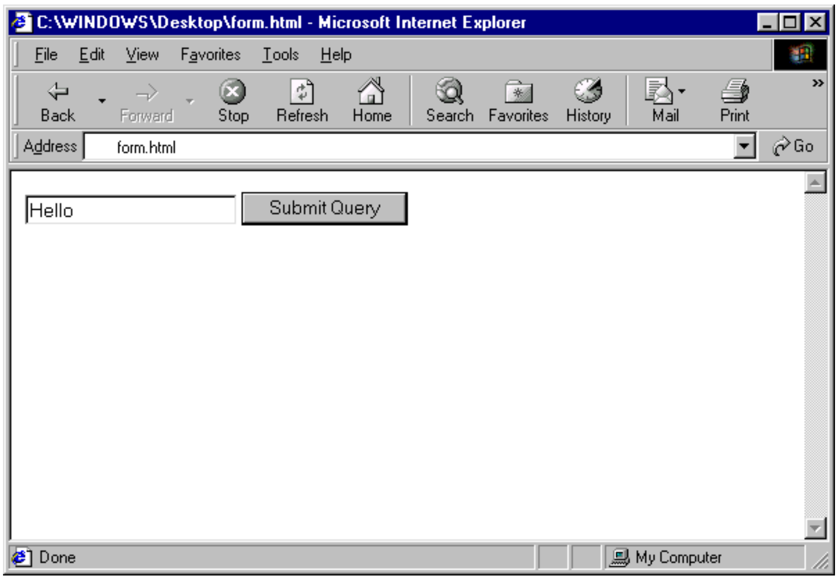
נתוני טופס (FORM) נשלחים אל השרת באחת משתי דרכים, post או get.

## Post

שיטה זו היא השיטה הטבעית לשליחת נתונים על גבי פרוטוקול HTTP. כדי לשלוח נתוני טופס ב- Post, יש להוסיף בשורת הגדרת הטופס את המילים method=post באופן הבא:

```
<form action=http://www.mysite.com method=post>
<input type=text name=x>
<input type=submit>
</form>
```

טופס זה ייראה כך בדפדפן:




בעת לחיצה על לחצן submit, המשתנה x יישלח לשרת המוגדר ב-action כשערכו יהיה המלל שהוזן לתוך שדה הטקסט. כלומר, אם המשתמש הזין לשדה את המלל 'Hello' ולחץ על לחצן Submit, המשתנה x וערכו יישלח לשרת www.mysite.com בצורה הבאה: x=Hello.

## Get

שיטה זו הינה שיטה חלופית לשליחת מידע לשרת והיא נבדלת משיטת Post בכך שכמות המידע שניתן להעביר בה היא מוגבלת וכן ניתן לראות את הנתונים בשורת הכתובת. כלומר, הנתונים יוצגו בהמשך לכתובת השרת. לדוגמה:

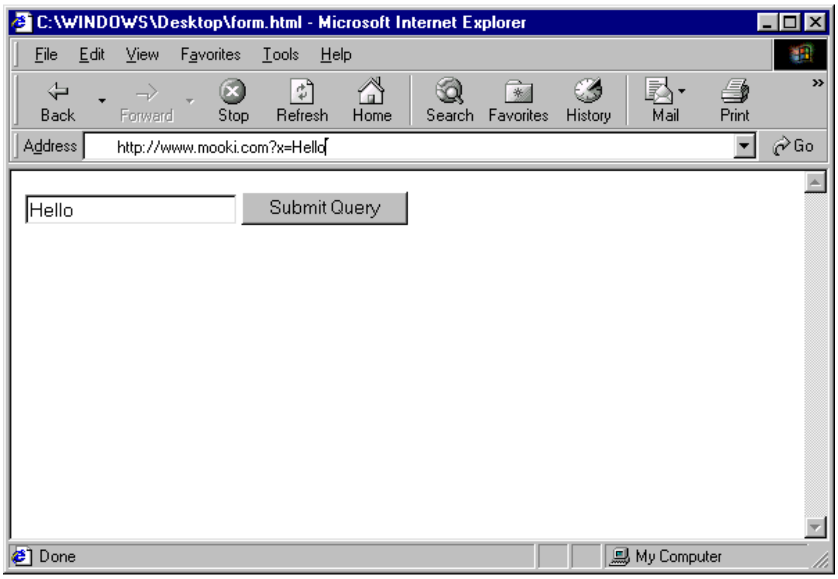
http://www.mooki.com?x=Hello

המשתנים וערכיהם מופרדים מכתובת השרת על ידי סימן שאלה (?). אם יש יותר ממשתנה אחד, יופרדו המשתנים על ידי סימן אמפרסנד (&) באופן הבא :

http://www.mooki.com?x=Hello&y=Kriger

כדי לשלוח את נתוני הטופס בשיטת Get, יש להוסיף לפקודת Form את המילים method=get, אם כי ברירת המחדל של הדפדפן היא לשלוח את הנתונים ב- Get אם לא נקבע אחרת.

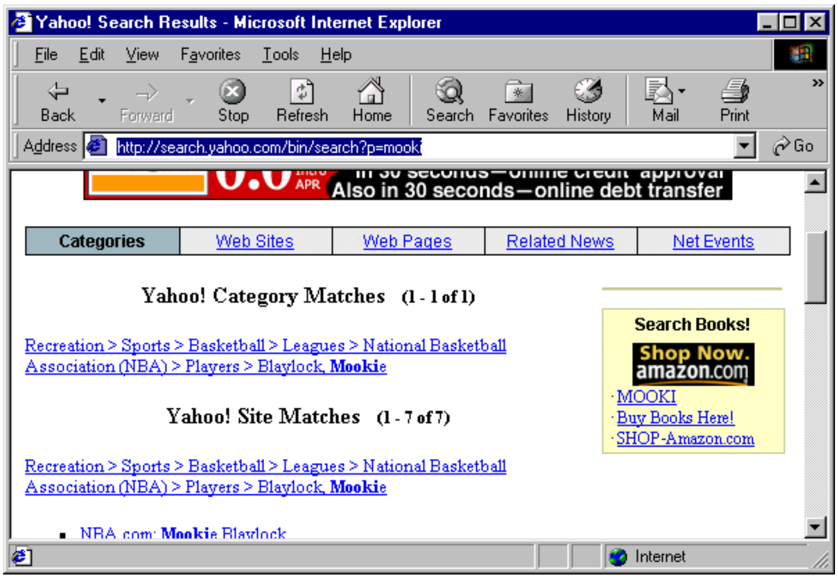
כך תיראה שליחת נתונים ב- Get. שימו לב לשורת הכתובת :




אם כן, קיימות שתי דרכים לשליחת הנתונים לשרת. לכל דרך יתרונות וחסרונות ויש להחליט לגבי כל יישום באיזו שיטה להשתמש. מנועי חיפוש מעדיפים להשתמש בשיטת Get, משום שעל שורת הכתובת המכילה את המשתנים המייצגים את החיפוש, ניתן להגדיר סימניה (Bookmark, Favorite) ובכך לשמור חיפושים מוצלחים.

לדוגמה, משתמש יוצר לשמור את הגדרת החיפוש הבאה:



# בניית מסמך HTML המאפשר חיפוש באתר של YAHOO

למעשה, ניתן לכתוב מסמך HTML המאפשר לשלוח שאילתות לאתר הבית של YAHOO.

היישום המבצע את החיפוש באתר הבית של YAHOO נקרא **search** והוא נמצא בכתובת:

http://search.yahoo.com/bin/search

יש להכניס כתובת זו לתכונת Action של הטופס, כך שמילות החיפוש יישלחו אליה, באופן הבא:

```
<form action="http://search.yahoo.com/bin/search">
```

לאחר מכן יש לבנות שדה טקסט בעל השם p, מכיון שמנוע החיפוש של YAHOO ממתין למשתנה בשם זה:

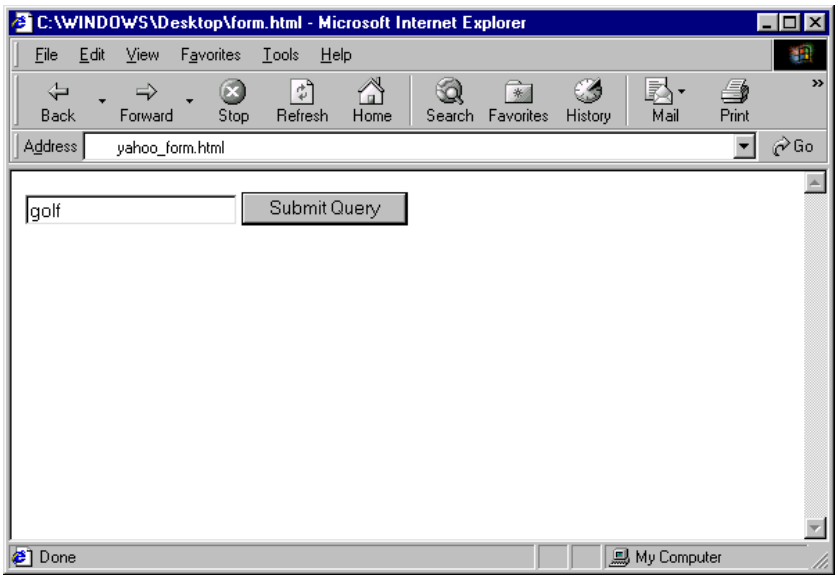
```
<input type=text name=p>
```

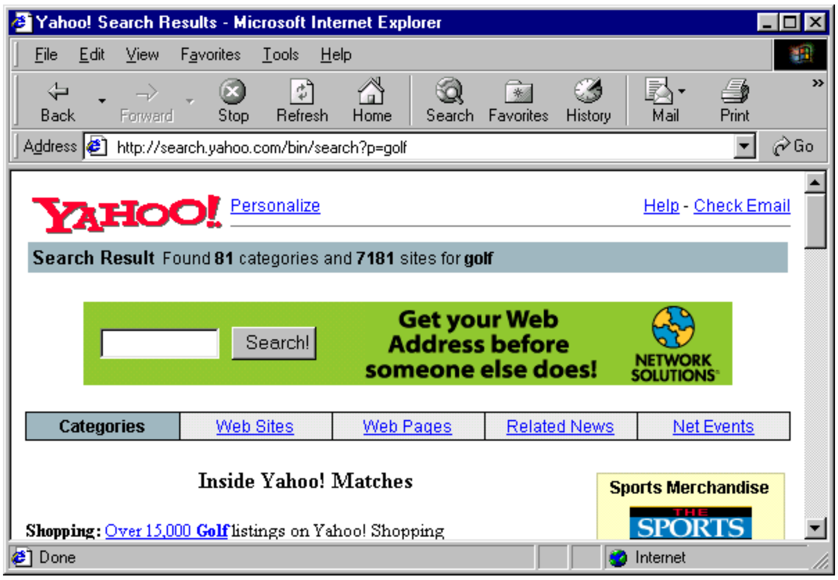
לפניך קוד HTML המלא השולח מילת חיפוש למנוע החיפוש של YAHOO:

yahoo_form.html

```
<form action=http://search.yahoo.com/bin/search method=get>
<input type=text name=p>
<input type=submit>
</form>
```

להלן המסכים:






כמובן, ניתן לייצר את שורת הכתובת באופן ידני כשמדובר בשיטה Get. לעומתה, מהווה השיטה Post אמצעי מאובטח יותר להעברת מידע, מכיון שלא ניתן לראות את המשתנים בשורת הכתובת.

# תרגיל ראשון בתקשורת בין שרת ללקוח

תחילה נבנה טופס HTML רגיל. בפקודת form נכניס ל- Action את כתובת השרת עליו אנו עובדים, כלומר **127.0.0.1** (ראה תרגיל בדיקת תקינות בפרק 2). אנו מעוניינים כי הנתונים שיוזנו לטופס יישלחו לקובץ ASP אותו נבנה בהמשך אשר ייקרא get_form.asp. אם כך, שורת הקוד הראשונה במסמך HTML תהיה:

<form action=http://127.0.0.1/get_form.asp method=get>

שימו לב כי נשתמש בשיטה Get.

נוסיף את המלל:

Type in your name:

לאחר מכן נוסיף שדה טקסט בשם x.

<input type=text name=x>

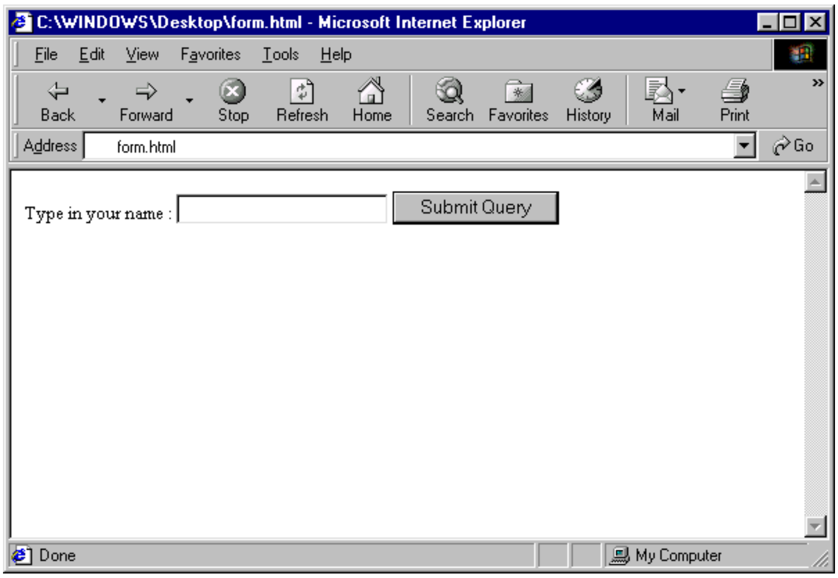
וכן נוסיף לחצן Submit ונסגור את הטופס באופן הבא:

<input type=submit>
</form>

כך ייראה הקובץ form.html :

```
<form action="http://127.0.0.1/get_form.asp" method=get>
Type in your name :
<input type=text name=x>
<input type=submit>
</form>
```

להלן המסך של form.html :




כעת נבנה את מסמך ASP.

לאובייקט Request מספר שיטות אשר יפורטו בהמשך, אך כרגע נתייחס לשיטה הראשונה : **QueryString()**. למעשה, מוגדרת QueryString כתכונה, אך בשל העובדה שיש להעביר אליה ערכים, נתייחס אליה בספר זה כאל שיטה.

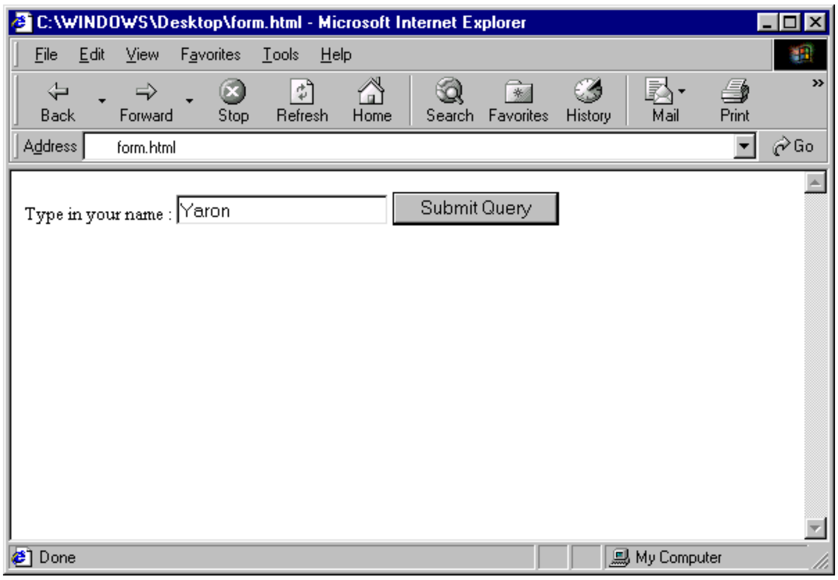
השיטה QueryString מקבלת את מחרוזת המשתנים הנשלחת בשיטה Get. יש לציין איזה משתנה אנו מעוניינים לקבל, ומכיון שהמשתנה הנשלח מטופס HTML שבנינו נקרא x, אם נרצה לקבל את ערכו בטופס HTML נצטרך לכתוב :

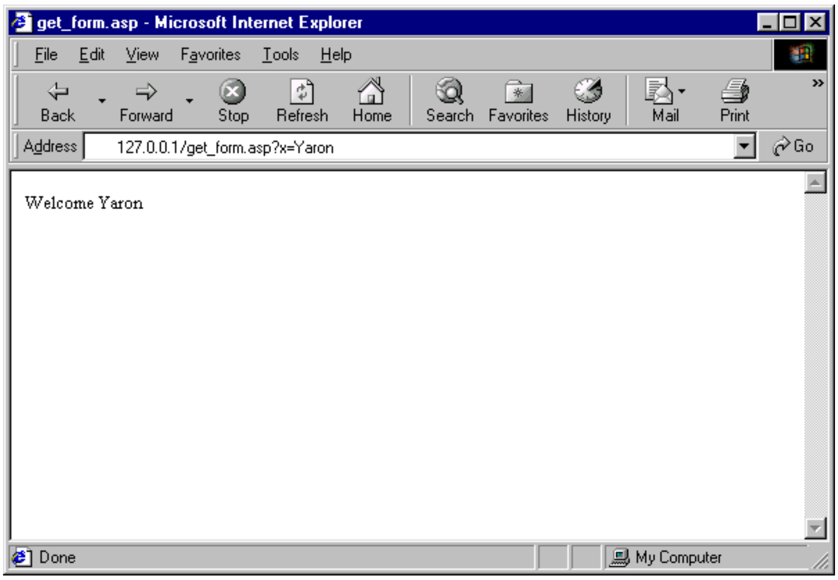
```
request.querystring("x")
```

בהדגמה פשוטה זו נכתוב קובץ ASP בשם get_form.asp, ובו נכתוב ברכת Welcome אליה נצרף את ערך המשתנה x אשר הגיע מהטופס באופן הבא :

```
Welcome <%=Request.QueryString("x")%>
```

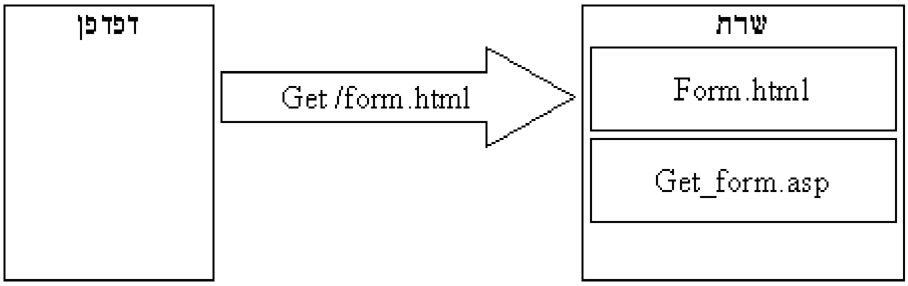
להלן המסכים של התרגיל:



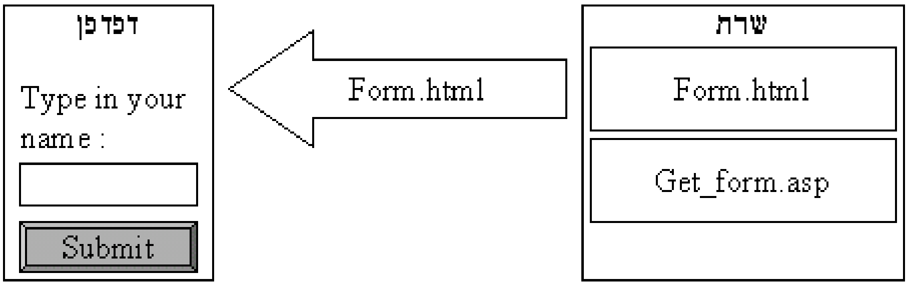
הבה נבין את התהליכים המתרחשים בתרגיל זה :

**שלב א**



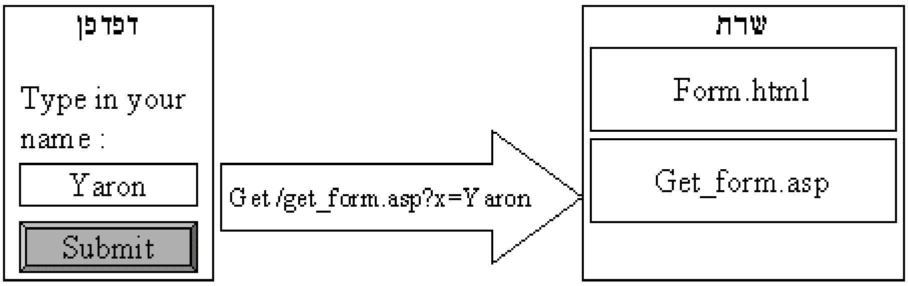

הדפדפן מבקש את מסמך HTML (form.html).

**שלב ב**



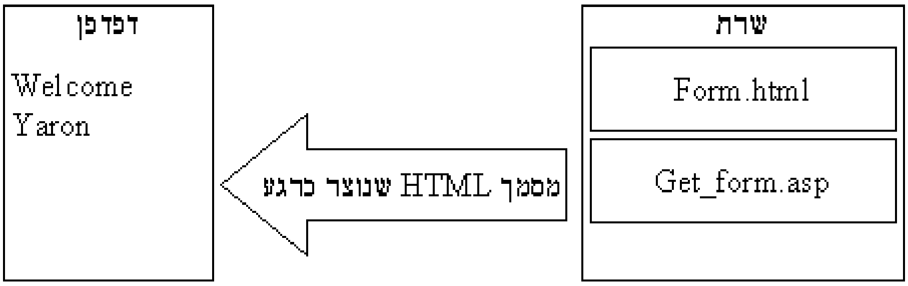

השרת שולח את מסמך HTML ללקוח. הדפדפן מבצע את קוד HTML שקיבל ומציג מלל, שדה טקסט ולחצן Submit.

**שלב ג**




המשתמש מקליד את שמו בשדה הטקסט ולוחץ על לחצן Submit. ברגע הלחיצה, אוסף הטופס את ערך השדה, משרשר אותו לכתובת המופיעה בהגדרת Action ושולח.

**שלב ד**




קובץ ASP מקבל את הפנייה ומכין מסמך HTML חדש, המורכב מהמלל Welcome ומערך המשתנה x. המסמך החדש נשלח שוב ללקוח.

# כרטיס ברכה אישי ב- ASP

קטעי הקוד הבאים יאפשרו לגולש להגדיר לעצמו כרטיס ברכה אשר יכלול צבעים שונים, גודל וצבע גופנים ואף תוכן מילולי דינמי.

הצעד הראשון יהיה לבנות מסמך HTML ראשוני לו נקרא personal_card.html.

נפתח מסמך HTML חדש :

```
<html>
<body bgcolor=teal text=white>
<center><font size=7>Personal Greeting Card Generator</font></center>
<hr>
Welcome to the personal greeting card generator. In this application you can
create<br>
Your own greeting card by determining the background color, font color & size and <br>
The greeting's message.<hr>
```

לאחר מכן, נבנה טופס אשר ישלח את הנתונים לקובץ ASP שייבנה בשלב הבא, וייקרא create_card.asp באופן הבא :

```
<form action=http://127.0.0.1/create_card.asp method=get>
```

כעת נוסיף שדות אשר יאפשרו לגולש להגדיר את צבע הרקע של כרטיס הברכה, צבע וגודל המלל ותוכן הברכה.

```
Choose a background color :
<input type=text name=bgcolor><br>
Choose a font color :
<input type=text name=fontcolor><br>
```

```
Choose a font size :
<select name=fontsize>
	<option value=1>1
	<option value=2>2
	<option value=3>3
	<option value=4>4
	<option value=5>5
	<option value=6>6
	<option value=7>7
</select><br>
<textarea name=cardmessage>
</textarea><br>
```

לאחר מכן, נוסיף לחצן Submit ונסגור את הטופס.

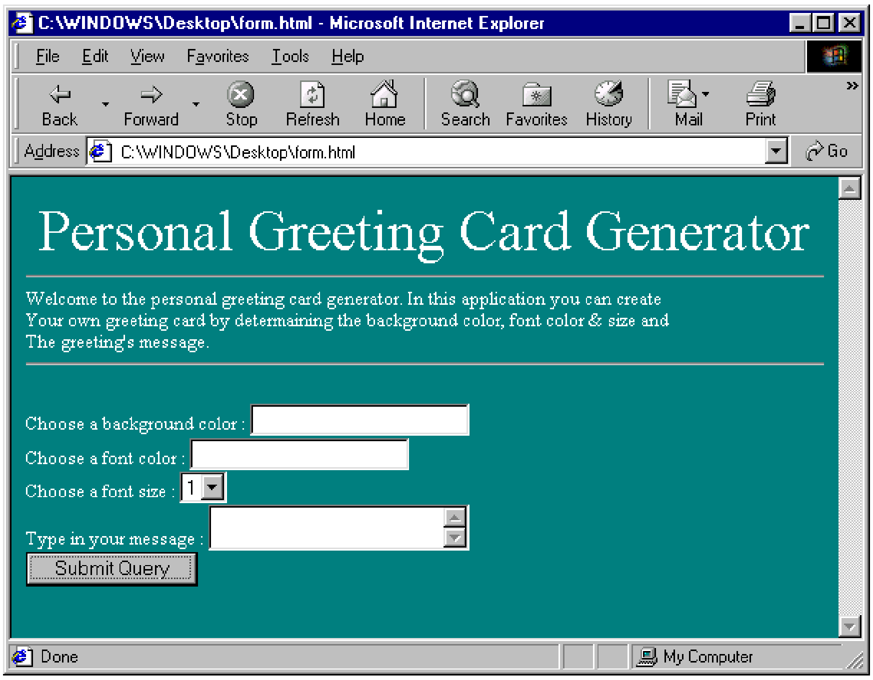
```
<input type=submit>
</form>
```

כך ייראה הקוד המלא של personal_card.html :

```
<html>
<body bgcolor=teal text=white>
<center><font size=7>Personal Greeting Card Generator</font></center>
<hr>
Welcome to the personal greeting card generator. In this application you can
create<br>
Your own greeting card by determining the background color, font color & size and <br>
The greeting's message.<hr>
<form action="http://127.0.0.1/create_card.asp" method=get>
Choose a background color :
<input type=text name=bgcolor><br>
Choose a font color :
<input type=text name=fontcolor><br>
Choose a font size :
<select name=fontsize>
	<option value=1>1
	<option value=2>2
	<option value=3>3
	<option value=4>4
	<option value=5>5
	<option value=6>6
	<option value=7>7
</select><br>
```

```
Type in your message :
<textarea name=cardmessage>
</textarea><br>
<input type=submit>
</form>
</html>
```

וכך נראה המסך שלו:



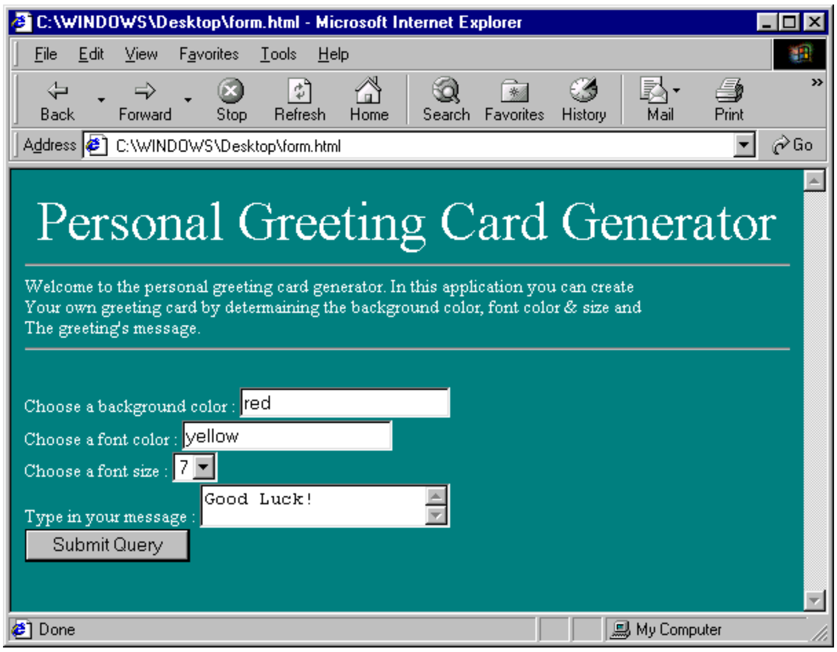

כעת, נכתוב את קובץ ASP בשם create_card.asp אשר יקבל את הנתונים מהטופס ויבנה באופן דינמי את כרטיס הברכה עבור המשתמש. בקובץ זה נשתמש ב- request.querystring עבור כל אחד מהפרמטרים שיישלחו מהטופס. להבהרה, אם הטופס ימולא בערכים הבאים: red, yellow, 7, Good Luck!, תישלח השורה הבאה אל השרת:

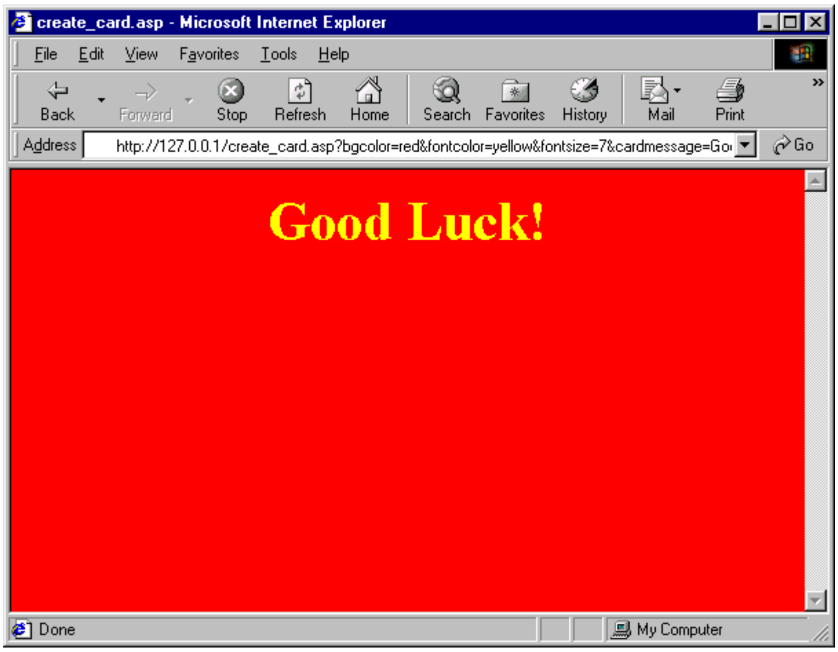
http://127.0.0.1/create_card.asp?bgcolor=red&fontcolor=yellow&fontsize=7&cardmessage=Good Luck!

כעת, כל שנותר הוא לשלב את פקודות HTML עם ערכי הטופס באופן הבא. להלן התוכן של הקובץ create_card.asp:

```
<body bgcolor=<%=request.querystring("bgcolor")%>
text=<%=request.querystring("fontcolor")%>>
<center>
<font size=<%=request.querystring("fontsize")%>>
<%=request.querystring("cardmessage")%>
```

להלן המסכים של יישום זה בפעולה:



# עוד על אובייקט Request

אנו יודעים אם כן, לשלוח נתונים לקובץ ASP ולהשתמש בהם. כעת נכיר את אובייקט Request לעומק. לאובייקט זה, כמו לאובייקטים אחרים, יש מספר שיטות ומאפיינים בהם ניתן להשתמש.

השיטה הראשונה בה השתמשנו היא ()querystring. שיטה זו כאמור משמשת לקבלת ערכי משתנים המגיעים מטופס בשיטת Get. אם הנתונים נשלחים בשיטת Post ניאלץ להשתמש בשיטה המקבילה Form. אין הבדל בין הפונקציונליות של שתי השיטות.
לקבלת משתנה x שנשלח ב- Get נכתוב:

```
request.querystring("x")
```

ולקבלת אותו משתנה שנשלח ב- Post נכתוב:

```
request.form("x")
```

## cookies

תכונה זו של אובייקט Request מאפשרת שליפה מקבצי Cookies של הלקוח, ועל כך ידון בפרק העוגיות.

## ServerVariables

תכונה נוספת של אובייקט request (אשר גם אליה נתייחס כשיטה), היא ServerVariables המאפשרת לשאוב 43 פריטי מידע על הלקוח ועל השרת.

לדוגמה, ניתן לקבל את כתובת IP של הגולש על ידי כתיבת הקוד הבא:

```
<%
Your IP is <%=request.servervariables("remote_addr")
%>
```

להלן חלק מהנתונים שניתן לקבל בעזרת request.servervariables:

- **request.servervariables("all_http")**

תכונה זו מאפשרת גישה ל- HTTP Header הנשלח מהלקוח. HTTP Header הוא הקדמה למסמך HTML הנשלח ללקוח וכן למידע המגיע ממנו, ובו מוגדרים נתונים כגון השפה בה משתמש הלקוח, הדפדפן הספציפי ועוד. ב- HTTP Header ניתן למצוא מידע חשוב על הלקוח הגולש. להלן דוגמה ל- HTTP Header:

```
HTTP_ACCEPT:*/*
HTTP_ACCEPT_LANGUAGE:he
HTTP_CONNECTION:Keep-Alive
HTTP_HOST:127.0.0.1
HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)
HTTP_COOKIE:ASPSESSIONIDFFEKPMTT=BPHPPECCPDMAHCEHIJCHIDEB
HTTP_ACCEPT_ENCODING:gzip, deflate
```

❖ **Request.servervariables("appl_physical_path")**

תכונה זו מאפשרת לקבל את התיקיה האמיתית בה נמצאים הקבצים.

❖ **Request. servervariables ("content_length")**

תכונה זו תחזיר את אורך המסמך שהתקבל מהלקוח.

❖ **Request. servervariables ("request_method")**

תכונה זו קובעת אם הגולש שולח את הנתונים ב- Post או ב- Get.

# ערך מוסף – לולאת for each

לולאה מיוחדת בה ניתן להשתמש היא לולאת for each. לולאה זו בנויה לרוץ על רשימת עצמים מוגדרת מראש ולהתייחס לכל פריט בנפרד.

לדוגמה, אם נפנה אל מסמך ASP ונשלח ב-GET את הפרמטרים הבאים: x=1&y=2&z=3, נוכל לקבוע לולאת for each אשר תחלץ את הערך מכל משתנה.

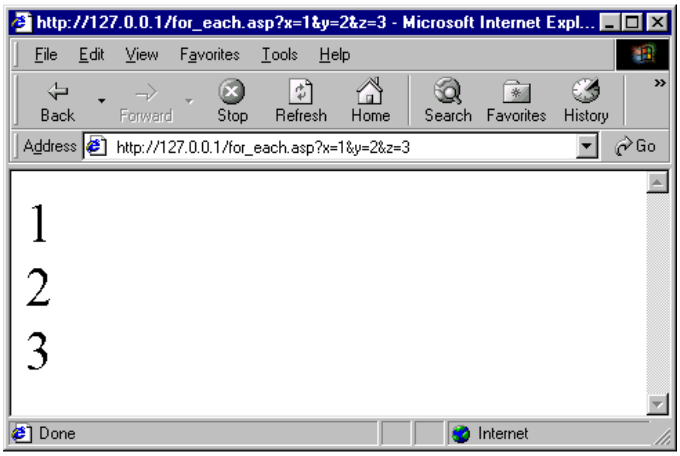
לדוגמה, התחביר של קובץ start_for_each.html:

```
<form action=for_each.asp method=get>
<input type=text name=x value=1><br>
<input type=text name=y value=2><br>
<input type=text name=z value=3><br>
<input type=submit>
</form>
```

להלן התחביר של קובץ asp בשם for_each.asp, הקובע כי עבור כל פריט (item) במחרוזת (querystring) יוצג ערכו:

```
<font size=7>
<%
for each item in request.querystring
%>
<%=request.querystring(item)%><br>
<%
next
%>
```

אם נקרא לקובץ זה עם הפרמטרים בשורת הכתובת (שימו לב לשורת הכתובת), נקבל את התוצאה הבאה:



## פרק 5

# אובייקט Response

## מהו אובייקט ASP?

ל- ASP3.0 7 אובייקטים. אובייקטים אלה מכילים קטעי קוד מוכנים המאפשרים למפתח לבצע פעולות מורכבות בשורות קוד בודדות. כפי שנלמד בפרק הקודם, כדי לקבל משתנה הנשלח מטופס יש לכתוב request.querystring("x") בלבד. כך ניתן להשתמש בכל אחד מהאובייקטים לביצוע מטלות שונות. האובייקטים המוגדרים ב- ASP הם :

request

response

application

session

server

object context

asp error

בפרק זה נכיר את אובייקט Response.

## Response

כשם שאובייקט Request מאפשר קבלת נתונים, משמש אובייקט Response לשליחת נתונים אל הלקוח. בפרק זה נסקור חלק מהשיטות והתכונות של אובייקט זה.

# Write

השיטה הראשונה אותה נסקור היא השיטה Write, המאפשרת כתיבה לקובץ HTML הנשלח אל הלקוח. כלומר, במקום לרשום:

```
<%
dim x
x="hello"
%>
Welcome <%=x%>
```

ניתן לרשום:

```
<%
dim x
x="hello"
response.write "Welcome " & x
%>
```

## שרשור

שימו לב, ששרשור המחרוזת מתבצע בעזרת סימן האמפרסנד (&). כשם שב-JavaScript אנו משרשרים מחרוזות טקסט בעזרת סימן החיבור (+), נשרשר ב- ASP בעזרת &. ניתן לשרשר ב- ASP גם בעזרת סימן החיבור, אך בשל מצבים אשר מכילים מספרים ובהם עלול סימן החיבור לבצע חיבור מתמטי, נשתמש באמפרסנד בספר זה.

להבנה מלאה של Response.write, מומלץ לנסות לבצע את תרגיל כרטיס הברכה מהפרק הקודם בשיטה זו, ולשנות אותו לפי הרשום בהמשך.

כך ייראה הקוד personal_card.html:

```
<html>
<body bgcolor=teal text=white>
<center><font size=7>Personal Greeting Card Generator</font></center>
<hr>
Welcome to the personal greeting card generator. In this application you can
create<br>
Your own greeting card by determining the background color, font color & size and <br>
The greeting's message.<hr>
<form action="http://127.0.0.1/create_card.asp" method=get>
Choose a background color :
<input type=text name=bgcolor><br>
Choose a font color :
<input type=text name=fontcolor><br>
```

```
Choose a font size :
<select name=fontsize>
            <option value=1>1
            <option value=2>2
            <option value=3>3
            <option value=4>4
            <option value=5>5
            <option value=6>6
            <option value=7>7
</select><br>
Type in your message :
<textarea name=cardmessage>
</textarea><br>
<input type=submit>
</form>
</html>
```

create_card.asp

```
<%
response.write "<body bgcolor=" & request.querystring("bgcolor")
response.write "text=" & request.querystring ("fontcolor") &" >"
response.write "<center>"
response.write "<font size=" & request.querystring("fontsize") & ">"
response write request.querystring("cardmessage")
%>
```

ניתן גם לחולל קוד זה בעזרת response.write אחד על ידי שימוש בקו תחתי לשבירת שורה, כפי שנלמד בפרק 3, תחביר ושורות קוד ראשונות, באופן הבא:

create_card3.asp

```
<%
response.write "<body bgcolor=" & request.querystring("bgcolor") &_
 " text=" & request.querystring ("fontcolor") &" >"  &_
 "<center>" &_
 "<font size=" & request.querystring("fontsize") & ">"  &_
 request.querystring("cardmessage")
%>
```

## Redirect

שיטה זו מאפשרת להפנות לקוח לאתר או דף אחר. בדוגמה הבאה, מזדהה הלקוח דרך טופס טקסט. במידה והסיסמה היא "eggnog", מופנה הלקוח לדף users.html. בכל מקרה אחר, יקבל הלקוח את עמוד הפתיחה.

תחילה נבנה טופס HTML:

```
authentication.html

<html>
<title></title>
<body bgcolor=teal text=white>
Please type in your password name:
<form action=http://127.0.0.1/check_authentication.asp method=post>
<input type=password name=p>
<input type=submit>
</form>
</html>
```

כעת נבנה קובץ ASP המקבל את הערך של p ובודק אם הוא שווה ל- eggnog. במידה וכן, תתבצע הפניה לדף users.html.

```
check_authentication.asp

<%
if request.form("p")="eggnog" then
response.redirect "users.html"
else
response.write "<body bgcolor=yellow><font size=4>Welcome to the guest area"
end if
%>
```

נוסיף גם את המסמך users.html כדי לסיים את ההדגמה:

```
users.html

<body bgcolor=pink>
<font size=5>
Welcome true user!
```

חשוב לציין כי פעולת Redirect יכולה להתבצע רק לפני שנשלח משהו ללקוח. כלומר, לא ניתן לבצע הפניית Redirect אם התבצע response.write כלשהו או נכתב קוד HTML (למעט מקרה של שימוש בתכונה Buffer אשר נידונה בהמשך פרק זה).

## Cookies

תכונה זו מאפשרת שליחת cookies ונידונה בהרחבה בפרק 8, על העוגיות.

## End

ברגע שמנוע ASP מזהה את המשפט response.end הוא מפסיק את שליחת הנתונים ללקוח. כלומר, בדוגמה הבאה לא תתבצע השורה השלישית.

```
<%
response.write "Will appear"
response.end
response.write "Will not appear"
%>
```

## Page Reentry

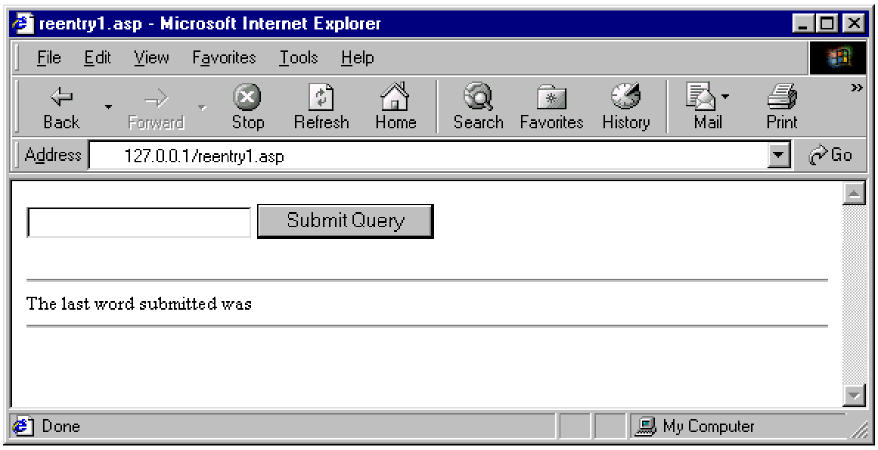
זו נקודה טובה להציג את אחת התפיסות המקובלות במיקרוסופט לכתיבת קוד. אם אינך בטוח בשליטתך בחומר, אין צורך לקרוא את הפסקאות הבאות משום שניתן להתבלבל בשלבי ההיכרות הראשונים עם ASP. תוכל לחזור לקטע זה בהמשך.

בכל הטפסים שכתבנו עד כה הגדרנו את יעד שליחת הנתונים ב- Action. אם לא נגדיר יעד לשליחת הנתונים, יישלחו הנתונים לאותו הקובץ. כעת, ניצור קובץ ASP אשר מציג טופס HTML וגם יודע לקבל את פרטיו. בשלב ראשון יוצג הטופס בכל פעם שעולה הדף.

Reentery1.asp

```
<form>
<input type=text name=x>
<input type=submit>
</form>
<hr>
The last word submitted was <%=request.querystring("x")%>
```

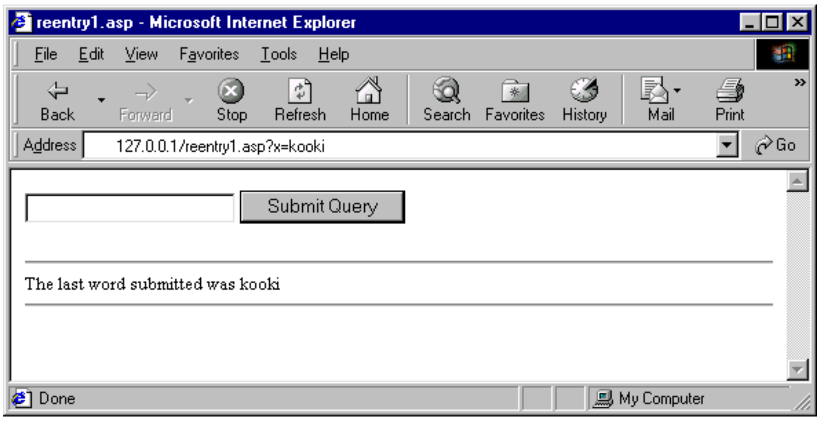
מסמך זה יעלה בפעם הראשונה כך:



שימו לב, שמכיון שאין עדיין ערך לשדה x, אין מילה כלשהי שמופיעה לאחר המשפט:

The last word submitted was

כעת נמלא את השדה ונלחץ על לחצן Submit. מכיון שאין הגדרת יעד לשליחת הטופס, יישלחו הנתונים שוב ל- Reentry1.asp והתוצאה תיראה כך:




השימוש המעשי של תפיסה זו הוא לרכז את הטופס ואת התגובה לטופס באותו מסמך.

בקובץ הבא נגדיר טופס HTML:

```
<form>
Type in your name :<input type=text>
<input type=submit>
</form>
```

מכיון שאין אנו מעוניינים להציג את הטופס שוב לאחר שליחת הנתונים, נבדוק אם הנתונים הוגשו על ידי אמצעי הבדיקה isEmpty.

התחביר הוא כזה:

```
if isempty(request.querystring("x")) then
```

בדיקה זו מבהירה האם הדף עולה בפעם הראשונה או בפעמים הבאות. אם הדף עולה בפעם הראשונה, אין עדיין ערך לשדה (isEmpty) ואילו לאחר לחיצה על לחצן Submit יהיה תמיד ערך למשתנה x.

אם כן, במידה והמשתנה חסר ערך (כלומר הדף עלה בפעם הראשונה), נציג את הטופס ונסיים את שליחת המשך הדף על ידי Response.end.

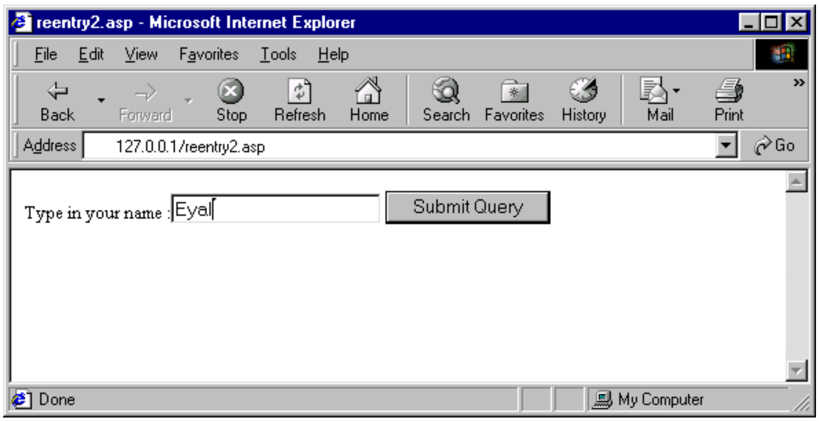
```
<%
if isempty(request.querystring("x")) then
%>
<form>
Type in your name :<input type=text>
<input type=submit>
</form>
<%
response.end
end if
%>
```

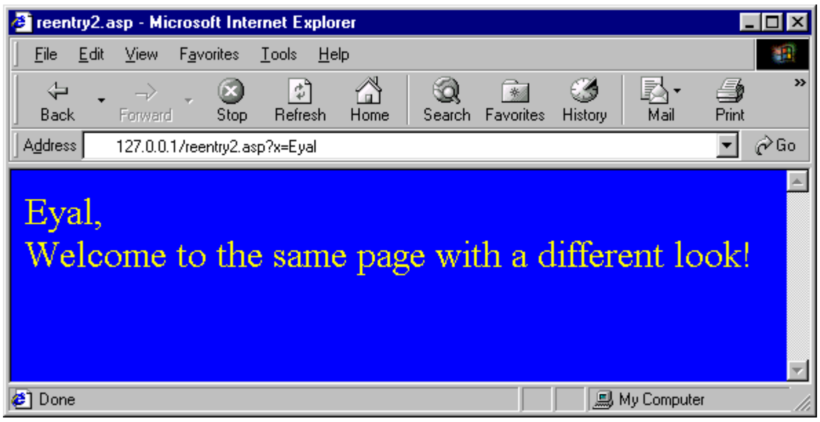
אם התנאי אינו מתקיים, כלומר, יש ערך למשתנה x, נבצע קטע קוד שונה באופן הבא:

reentry2.asp

```
<%
if IsEmpty(Request.QueryString("x"))then
%>
<form>
        Type in your name:
        <input type=text name=x>
        <input type=submit>
        </form>
<%
Response.End
end if%>
<body bgcolor=blue text=yellow>
<font size=5>
 <%=Request.QueryString("x")%>,
 <BR>
Welcome to the same page with a different look!
```

כך ייראו המסכים בפעם הראשונה והשנייה:




# Expires

תכונה זו מגדירה את הזמן שלאחריו. כאשר ינסה המשתמש להיכנס שוב לאותו המסמך, יטען הדפדפן את המסמך מהשרת ולא מהזיכרון המקומי שלו. תכונה זו מקבלת ערכים בדקות. כלומר, אם נקבע שלמסמך מסוים 3 דקות בתכונת Expires, יימחק מסמך זה מזיכרון הדפדפן לאחר 3 דקות.

לדוגמה:

```
<%
response.expires=3
%>
```

ניתן להשתמש גם ב- Expiresabsolute כדי להגדיר תאריך תפוגה מוחלט לדף.

# Buffer

תכונה זו, המקבלת ערכי true או false, מגדירה למנוע ASP האם להתחיל ולשלוח פקודות HTML ללקוח לפני ניתוח כל הדף (response.buffer=false), או האם לסיים לנתח את כל המסמך ורק אז לשלוח ללקוח את התוצאה (response.buffer=true). הגדרת Buffer חייבת להופיע בתחילת מסמך ASP.

הערך הבסיסי של Buffer הוא False.

# AppendToLog

תכונה זו מאפשרת להוסיף שורות טקסט לקבצי LOG של שרת HTTP. ניתן להוסיף עד 80 תווים ללא שימוש בפסיקים. לדוגמה:

```
response.appendtolog("The user left this site after 2 minutes")
```

# AddHeader

תכונה זו מאפשרת להוסיף שדות ל- HTTP Header באופן הבא:

```
response.addheader "newfield","valuefornewfield"
```

## פרק 6

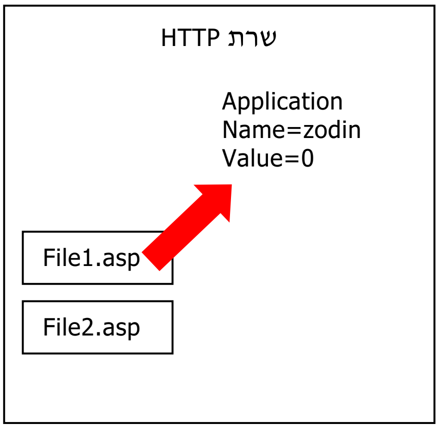
# אובייקט Application

לעיתים נוצר הצורך לשמור מידע גלובלי ביישום. הדוגמה הקלאסית לכך היא הצורך להוסיף לאתר מונה ביקורים (counter). מכיון שדפי ASP מתעוררים לחיים עם בקשה מהמשתמש ו"נסגרים" עם סיום ביצוע פעולתם, אין אפשרות לשמור ערכים כגון מספר המבקרים בדף ASP. לצורך כך, נוצר אובייקט Application המהווה אובייקט חיצוני ליישום המכיל ערכים ונגיש לכל דף ASP בשרת. באובייקט זה ניתן לאחסן ערכים קבועים או משתנים אשר ישמשו את היישום (מכאן נגזר השם Application). לצורך הבנת הרעיון העומד מאחורי השימוש באובייקט זה, נבחן את התהליכים המתבצעים בהוספת מונה ביקורים לאתר.

## הוספת מונה ביקורים (counter)

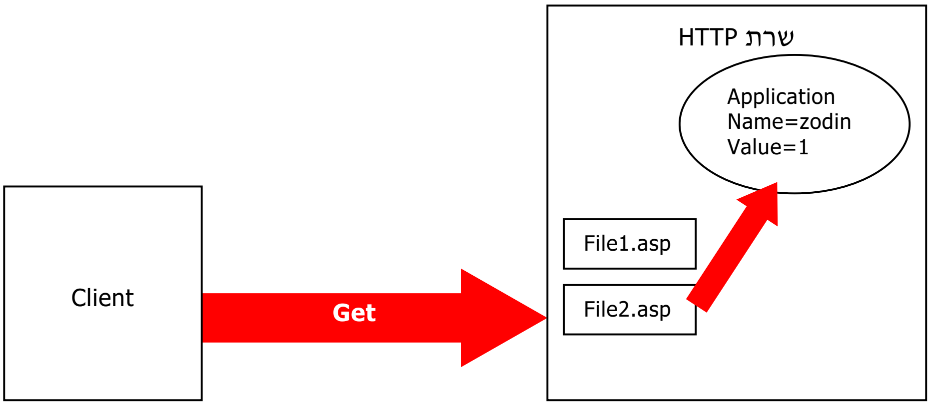
### שלב א

בשלב זה יוצר מסמך ASP אובייקט Application ומעניק לו את השם "zodin". יש צורך בשם עבור כל אובייקט Application כדי לאפשר מספר רב של משתנים מסוג זה. מייד עם יצירת האובייקט וקביעת שמו, קובע מסמך ASP את הערך ההתחלתי של האובייקט כ- 0.



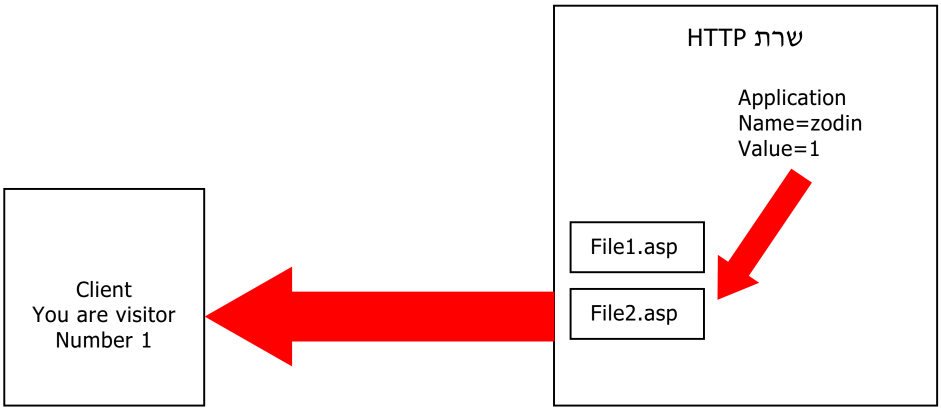
## שלב ב

משתמש נכנס לאתר ומבקש את אחד ממסמכי ASP. אותו מסמך ניגש ומעלה את ערכו של אובייקט Application הקרוי zodin ב- 1.




## שלב ג

מסמך ASP לוקח את ערכו החדש של zodin ושולח אותו ללקוח (משורשר עם המלל – you are visitor Number 1).




מרגע זה ואילך, כל מסמך ASP יוכל לבצע את אותו תהליך וכך יישמר מספר המבקרים באתר.

אובייקט Application ממשיך להתקיים עד לסגירת השרת או למחיקתו בצורה יישומית.

# יצירת אובייקט Application

תחביר יצירת אובייקט Application הוא פשוט. יש לכתוב את המילה Application ולאחריה את שם המשתנה אותו ניצור כגון "zodin". לקביעת ערך המשתנה נוסיף את הסימן שווה (=) ולאחריו את הערך הרצוי. לדוגמה:

```
application("zodin")=1
```

כדי לבצע את קטע הקוד הנ"ל, נכתוב אותו במסמך ASP ונפנה אליו פעם אחת. כדאי להוסיף מלל כלשהו כדי לקבל אינדיקציה שהקוד התבצע. לדוגמה:

```
application_create.asp

<%
application("zodin")=1
%>
The application variable has been set
```

כדי לראות את ערך המשתנה, נציג אותו באחת משתי הדרכים שנלמדו לשליחת מידע ללקוח:

```
The application variable "zodin" is set to <%=application("zodin")%>
```

או

```
<%
response.write "The application variable "zodin" is set to" & application("zodin")
%>
```

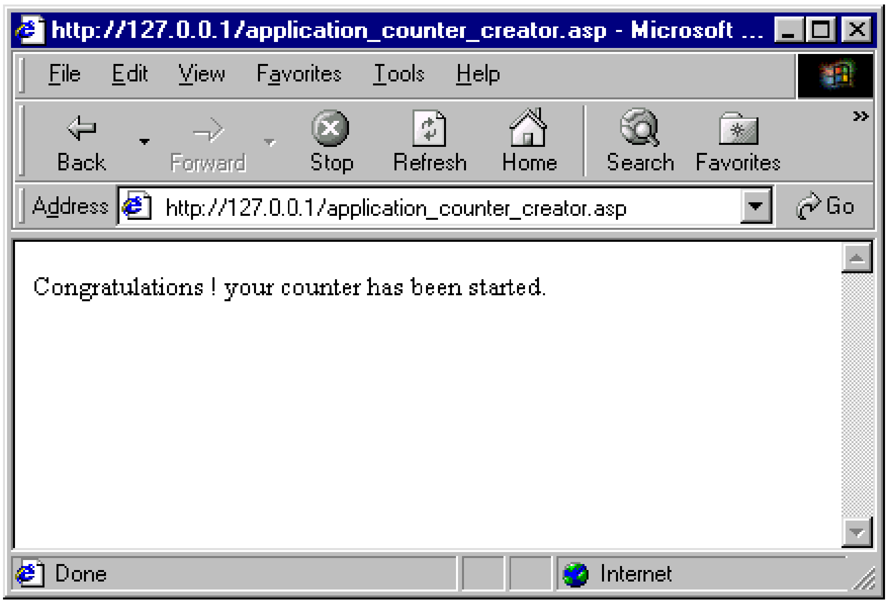
# בניית מונה ביקורים באתר

לצורך בניית המונה ואחסונו במשתנה מסוג Application, נבנה מסמך ASP אשר ייצור את המשתנה Excellent_counter.

```
Application_counter_creator.asp

<%
application("excellent_counter")=0
%>
Congratulations ! your counter has been started.
```

נריץ את קובץ ASP פעם אחת ונקבל את התשובה :

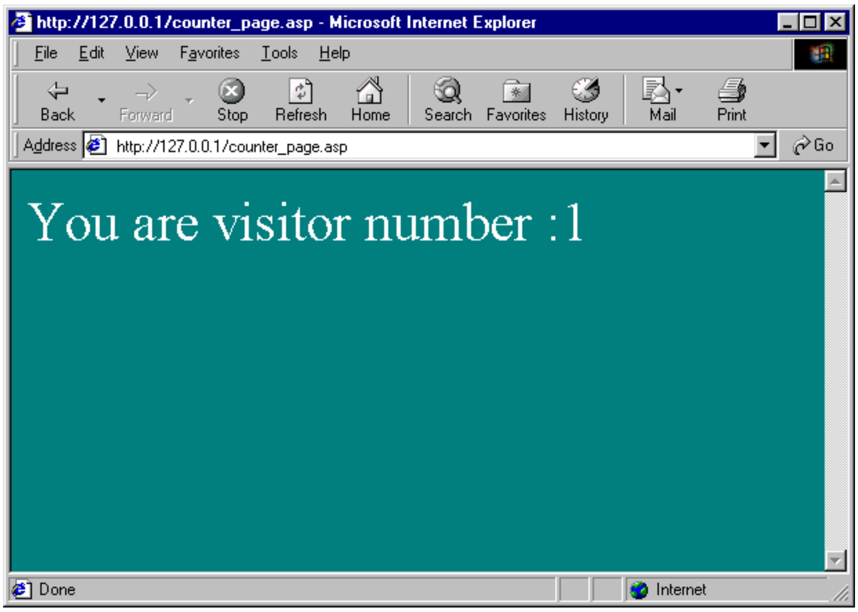



כעת, נבנה מסמך ASP אשר יעלה את ערך המשתנה excellent_counter ב- 1 ויציג את ערכו למשתמש. כמו כן, נשרשר את המשפט: "You are visitor number".

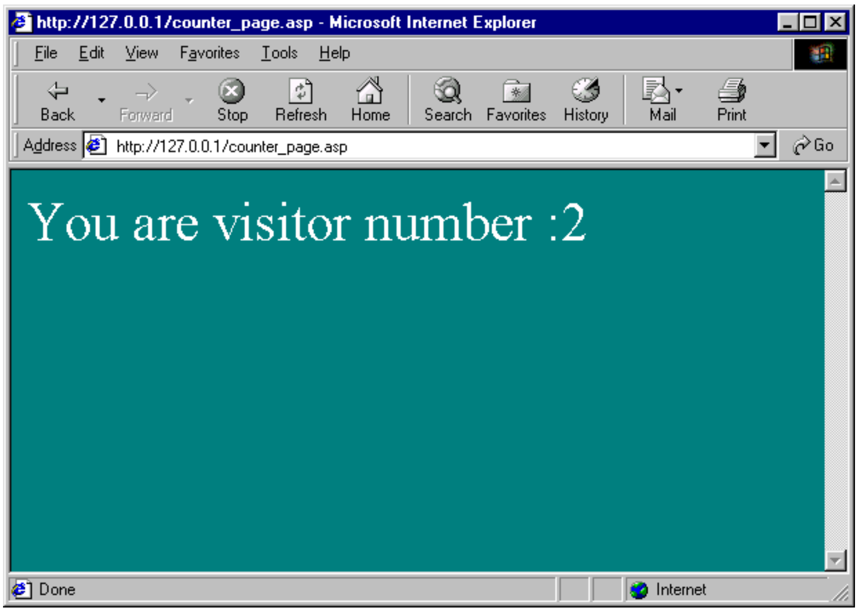
Counter_page.asp

```
<body bgcolor=teal text=white>
<font size=7>
<%
application("excellent_counter")=application("excellent_counter")+1
response.write "You are visitor number :" & application("excellent_counter")
%>
```

נפנה אל המסמך counter_page.asp בפעם הראשונה ונקבל את המסך הבא:




כאשר נבצע טעינה מחודשת של המסמך על ידי refresh או reload, נקבל:




בכל פעם שניגש למשתנה excellent_counter, נקבל את הערך שקבע הגולש הקודם.

# contents

לאובייקט Application תכונה המאגדת בתוכה את כל משתני Application שנוצרו. עם זאת, לא ניתן לגשת ל- Content באופן רגיל, אלא יש צורך להשתמש בלולאת for each (המוזכרת בפרק 4, אובייקט Request).

וכך, עבור כל פריט (item) ברשימת האובייקטים מסוג application (application.contents), יוצג ערך המשתנה באופן הבא:

application_contents.asp

```
<%
for each item in application.contents
response.write application.contents(item) & "<br>"
next
%>
```

זאת במידה וקיימים אובייקטים מהסוג הנ"ל.

# Contents.remove

שיטה זו, הנתמכת בגרסת asp 3.0 ומעלה, מאפשרת מחיקת אובייקט application ספציפי באופן הבא:

```
Application.Contents.Remove("zodin")
```

# Contents.removeall

שיטה זו, אף היא נתמכת מגירסה 3.0 ומעלה, מאפשרת מחיקת כל האובייקטים מסוג Application תוך שימוש בתחביר:

```
Application.Contents.RemoveAll( )
```

# בניית Chat פשוט באמצעות ASP

פרוטוקול HTTP הינו פרוטוקול חסר קשר רציף (stateless), ולכן קיימת בעיה עקרונית בבניית chat המחייב קשר רציף לכל המשתתפים בשיחה, מכיון שיש לשלוח עדכונים על משפטים חדשים שהוזנו. עם זאת, ניתן לדמות קשר רציף על ידי מנגנון השהייה הטוען שוב את הדף המעודכן מהשרת כל פרק זמן מסוים. זאת נבצע על ידי פונקציית JavaScript הנקראת **setTimeout**.

את ערכי המשתנים באובייקט Application יכולים לראות כל המשתמשים, ולכן נוכל להשתמש בו כדי לאחסן את פרוטוקול השיחה העדכני. נגדיר אובייקט Application שיצבור את המשפטים המוזנים מכל המשתתפים בשיחה, יפריד אותם באמצעות <br> ויאפשר לכולם לצפות בתוכנו.

נבנה frameset אשר יכיל חלק תחתון סטטי שיאפשר הזנת משפטים, וחלק עליון אשר יכיל את פרוטוקול השיחה ויתעדכן כל פרק זמן נתון.

ChatFrameSet.html

```
<frameset rows="70%,*" border="1">
<frame src="chat.asp" name="chatWin">
<frame src="input2.asp" name="inputWin">
</frameset>
```

כעת נבנה את החלק העליון המתעדכן כל 2 שניות ומדפיס את משתנה Chat של אובייקט מסוג Application (המשותף לכל המשתמשים) המכיל את פרוטוקול השיחה:

Chat.asp

```
<SCRIPT LANGUAGE=javascript>
function reloadPage()
{
	location.reload()
}
setTimeout('reloadPage()',2000)
</SCRIPT>
<%=application("chat")%>
```

נבנה את החלק התחתון המקבל את שם המשתתף בשיחה ואת המשפטים. חלק זה אף אחראי על הטיפול באובייקט Application. עם כל הזנה של מידע, ישרשר חלק זה שבירת שורה <br>, שם המשתמש שהזין את המשפט והמשפט עצמו (שימו לב כי נשתמש בתפיסת Page Reentry המופיעה בפרק 5, אובייקט Response).

Input.asp

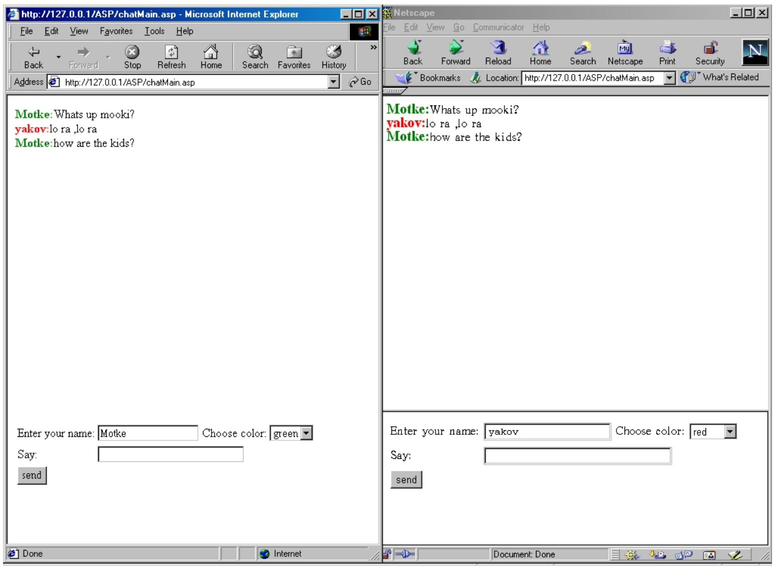
```
<%if not(isempty(Request.Form("say")))then
myName=Request.Form("myName")
iSaid = Request.Form("say")
application("chat")=application("chat")_
& "<B>" & myName & ":</b>"_
& iSaid & "<BR>"
end if%>
<form method=post>
<table>
Name<input type=text name=myName value="<%=myName%>"><br>
Say:<input type=text name=say size=30><br>
<input type=submit value="Talk"></form>
```

שיפור של chat יכלול צבעים שונים לכל משתתף באופן הבא :

ColorInput.asp

```
<%
colorArray = Array("red","green","blue")
if not(isempty(Request.Form("say"))) then
myName=Request.Form("myName")
myColor=Request.Form("userColor")
iSaid = Request.Form("say")
        application("chat")=application("chat")_
        & "<font color=" & myColor & "><B>"_
        & myName& ":</b></font>" & iSaid & "<br>"
end if%>
<form method=post>
<table>
Name<input type=text name=myName value="<%=myName%>">  
    Color<select name=userColor>
    <%for i=0 to 2
        if colorArray(i)=myColor then%>
        <option selected><%=colorArray(i)%></option>
        <%else%>
        <option><%=colorArray(i)%></option>
        <%end if
        next%></select>
Say<input type=text name=say size=30><br>
<input type=submit value="Talk">
</form>
```

כדי לבדוק את ה- Chat, ניתן לדמות מספר משתמשים על ידי פתיחת שני חלונות של דפדפנים. כך תיראה התוצאה:



```
Input.asp

<%if not(isempty(Request.Form("say")))then
myName=Request.Form("myName")
iSaid = Request.Form("say")
application("chat")=application("chat")_
& "<B>" & myName & ":</b>"_
& iSaid & "<BR>"
end if%>
<form method=post>
<table>
Name<input type=text name=myName value="<%=myName%>"><br>
Say:<input type=text name=say size=30><br>
<input type=submit value="Talk"></form>
```

שיפור של chat יכלול צבעים שונים לכל משתתף באופן הבא:

```
ColorInput.asp

<%
colorArray = Array("red","green","blue")
if not(isempty(Request.Form("say")))then
myName=Request.Form("myName")
myColor=Request.Form("userColor")
iSaid = Request.Form("say")
		application("chat")=application("chat")_
		& "<font color=" & myColor & "><B>"_
		& myName& ":</b></font>" & iSaid & "<br>"
end if%>
<form method=post>
<table>
Name<input type=text name=myName value="<%=myName%>">  
	Color<select name=userColor>
	<%for i=0 to 2
		if colorArray(i)=myColor then%>
		<option selected><%=colorArray(i)%></option>
		<%else%>
		<option><%=colorArray(i)%></option>
		<%end if
		next%></select>
Say<input type=text name=say size=30><br>
<input type=submit value="Talk">
</form>
```

## פרק 7

# אובייקט Session

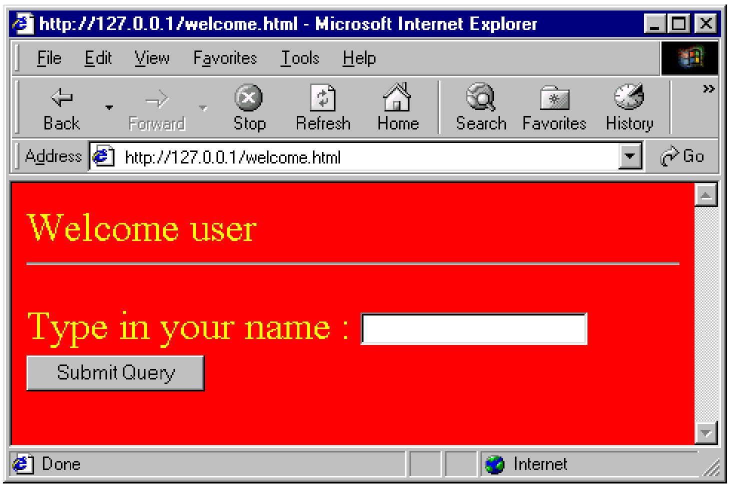
בפרק שעבר השתמשנו באובייקט Application כדי לשמור נתונים גלובליים ברמת היישום. בפרק זה נראה כיצד ניתן לשמור נתונים שונים עבור כל גולש. היישומים לשמירת נתונים עבור כל גולש הם רבים ונעסוק באחד החשובים בהמשך הפרק, אך תחילה ניישם רעיון פשוט המגדיר כי בכל דף באתר, יקבל המשתמש את שמו (זאת לאחר שהקיש אותו בדף הראשון).

נתחיל בבניית מסמך HTML המכיל טופס להזנת שם משתמש.

Welcome.html

```
<body bgcolor=red text=yellow>
<font size=6>
Welcome user<hr>
<form action=session1.asp>
Type in your name :
<input type=text name=n>
<br>
<input type=submit>
</form>
```

הדף הראשון ייראה כך :




לאחר מכן נבנה את מסמך ASP אשר יקבל את המשתנה n ויזין אותו אל משתנה מסוג session.

יצירת משתנה מסוג session הינה פשוטה ותואמת את התחביר ליצירת משתנה מסוג application, באופן הבא :

```
session("surfer_name")=.....
```

להלן הקוד :

```
session1.asp

<%
session("surfer_name")=request.querystring("n")
%>
<body bgcolor=blue text=white link=yellow vlink=yellow>
<font size=6>
Welcome <%=session("surfer_name")%>
<br>
<a href=session2.asp>next page</a>
```

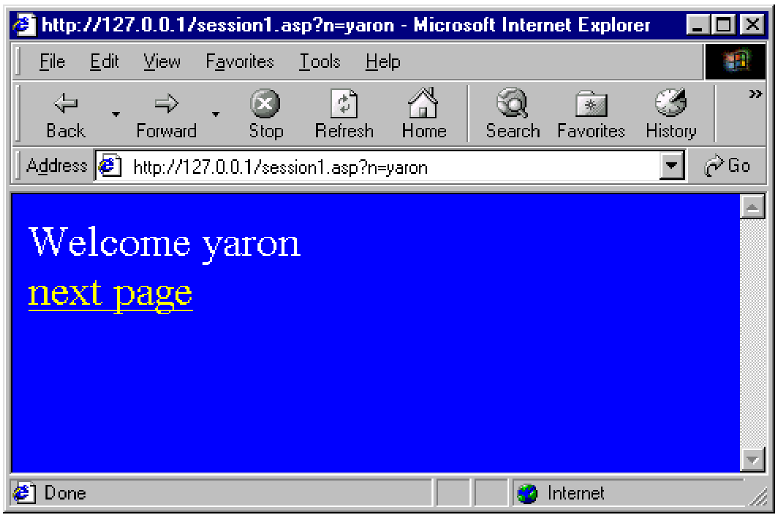
שימו לב כי קיים בקוד קישור לדף נוסף (session2.asp). מכיון ששם המשתמש נשמר באובייקט מסוג session, נוכל לקבל את שמו גם במסמכים נוספים כגון session2.asp כדלהלן :

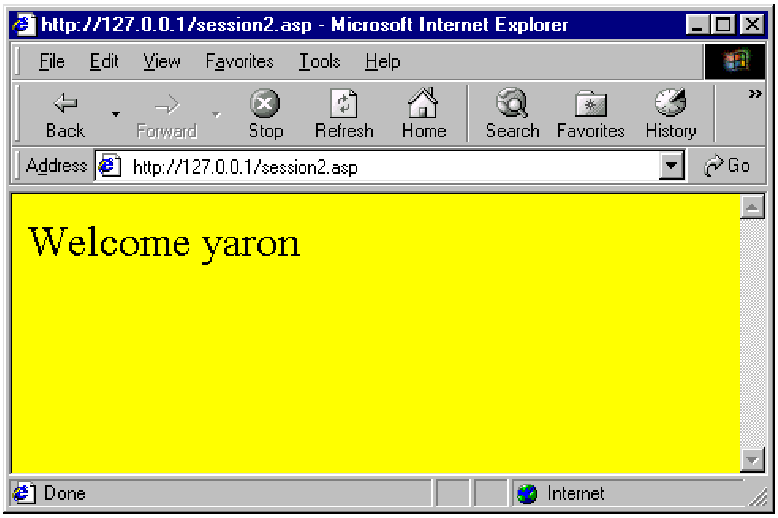
```
session2.asp

<body bgcolor=yellow>
<font size=6>
Welcome <%=session("surfer_name")%>
```

כמובן, כל משתמש יקבל את השם שהזין. ניתן לבדוק זאת על ידי כניסה למסמך משני דפדפנים שונים (זאת משום שאובייקט Session משתמש למעשה בעוגיות (cookies) ושומר את המידע על מספר ה- session אצל הלקוח).

כך ייראו שני המסמכים של session1.asp ו- session2.asp :




אם כך, ניתן לשמור מידע עבור כל לקוח. מכיון שהיישום מתבצע בעזרת עוגיות, ניתן לוותר על השימוש באובייקט זה ולקיים את הפונקציונליות שלו תוך שימוש "ידני" בעוגיות. אך מכיון שאובייקט Session מקצר תהליכים, מומלץ להשתמש בו במקרים בהם מתעורר צורך שכזה.

# אבטחת מידע

אחד השימושים העיקריים של עוגיות, ולכן גם של אובייקט session, הוא אבטחת דפים מסוימים. ניתן לבקש מהגולש להזדהות עבור מסמך, ולכן ניתן להגן על מידע על ידי בדיקת הסיסמה. אולם, בדרך כלל, אנו מעוניינים להגן על מספר רב של דפים, ולכן נוצרת בעיה.

אם נבקש סיסמה בכל דף, הרי שהפכנו את האתר ללא ידידותי בלשון המעטה, ואם נבקש סיסמה רק בדף הראשון המכיל קישורים לדפים נוספים, יוכל כל גולש לבצע גישה ישירה (בה הוא מציין את שם הדף בשורת הכתובת) אל כל דף שאינו מוגן.

כל שעלינו לעשות, הוא לאחסן את האישור (במידה וההזדהות עברה בהצלחה) למשתמש באובייקט session, ולבדוק אותו בצורה שקופה בכל דף.

נתחיל בבניית המסמך הראשון, אשר מבקש את שם המשתמש ומבצע בדיקה. אם הבדיקה נכונה, כלומר הוקשה  סיסמה הנכונה, יוכל המשתמש לראות את תוכן הדף. לצורך כך נשתמש בתפיסת Page Reentry (ראה פרק 5, אובייקט Response). כמו כן, נשתמש בשיטת post אשר אינה מראה את הפרמטרים בשורת הכתובת ומחייבת שימוש בשיטה request.form.

Authentication1.asp

```
<%
if isempty(request.form("secret_name")) then
response.write "<font size=5>Please enter your secret name:" &_
"<form method=post><input type=password name=secret_name>" &_
"<input type=submit> </form>"
response.end
else
    if request.form("secret_name")="muchraka" then
    session("secret")="ok"
    response.write "<body bgcolor=black text=white link=white vlink=white>" &_
    "<font size=5>Welcome " & _
    "<br>You can now access other secure pages like:<br>" &_
    "<a href=authentication2.asp>secure second page</a>"
    else
    response.write("Wrong password")
    end if
end if
%>
```

אם הסיסמה נכונה, מאחסן דף ASP את האישור ok במשתנה secret מסוג session, ומציג למשתמש את הדף עם הקישור לדף המאובטח הבא.

כעת, מכיון שרק למשתמש אשר הזין את הסיסמה הנכונה (muchraka), יש משתנה מסוג session המכיל את הערך ok, ניתן לבדוק בכל דף מאובטח נוסף כי באמת קיים אובייקט זה והאם ערכו הוא ok באופן הבא:

authentication2.asp

```
<%
if session("secret")="ok" then
response.write "<font size=5>Welcome authorized user!"
else
response.write "Access denied"
end if
%>
```

ניתן לבדוק את בטיחות האתר על ידי ניסיון גישה ל- 127.0.0.1/authentication2.asp על ידי דפדפן אחר או על ידי סגירה ופתיחה של הדפדפן (לצורך מחיקת העוגיה), ללא מעבר דרך authentication1.asp. התשובה שתתקבל היא: Access denied.

# Timeout

אובייקט session נוצר עבור כל משתמש ומשמש לתחימת ביקור באתר (session), ולכן יש הגבלת זמן המגדירה כי אם המשתמש לא ביצע כל פעילות במהלך 20 דקות, נמחק אובייקט session. אותן 20 דקות הן ברירת מחדל וניתנות לשינוי על ידי התכונה timeout באופן הבא:

```
Session.Timeout = 12
```

הערכים המתקבלים הם בדקות.

# Abandon

כמו כן, ניתן לסיים session באופן יזום על ידי השיטה abandon באופן הבא:

```
Session.Abandon
```

עם השימוש בשיטה זו, נמחקים כל האובייקטים המוגדרים כ- session וכל ההגדרות המוגדרות עבור משתמש.

## Contents

שיטה זו הנתמכת בגירסה 3.0 ומעלה, דומה לשיטה הזהה באובייקט Application, ומאפשרת גישה לכל האובייקטים המוגדרים כ- Session על ידי שימוש בלולאת for each באופן הבא:

```
for each item in session.contents
response.write session.contents(item) & "<br>"
next
```

## contents.remove

שיטה זו, הנתמכת בגירסה 3.0 ומעלה, דומה לשיטה הזהה באובייקט Application. ניתן למחוק אובייקט session מסוים על ידי השימוש בתחביר:

```
session.contents.remove("pooki")
```

## contents.removeall

שיטה זו, הנתמכת בגירסה 3.0 ומעלה, דומה לשיטה הזהה באובייקט Application ומאפשרת מחיקת כל האובייקטים המוגדרים כ- Session עבור המשתמש באופן הבא:

```
session.contents.removeall()
```

# פרק 8

# עוגיות (cookies)

ספר זה מיועד לכותבי JavaScript, ולכן לא נדון כאן במהות העוגיות, אלא נבחן את הכלים שמציע ASP לטיפול בסוג זה של אחסון ואחזור נתונים.

ניתן לשתול cookies באמצעות response.cookies ולשלוף באמצעות request.cookies. חשוב לציין, כי למעט מקרים בהם התכונה response.buffer מקבלת ערך True, יש לבצע כל פעילות הקשורה ל- cookies לפני כל סוג של קוד HTML הנשלח לשרת, כלומר, בשורות הראשונות של קובץ ASP.

כידוע, לכל Cookie מבנה ברור המוגדר במחרוזת טקסטואלית ובה:

- ❖ שם Cookie אשר לפיו ניתן לגשת אליה.
- ❖ ערך Cookie המאפשר אחסון מידע.
- ❖ תאריך תפוגת Cookie אשר אם לא קיים, מגביל את קיום ה- Cookie עד סגירת הדפדפן.
- ❖ מסלול מורשה הקובע אם למסמכים נוספים על השרת מותר לגשת לאותם Cookies.

בשונה מ- JavaScript, בה מוגדרת Cookie בשורה אחת המופרדת בנקודה-פסיק (;), ב-ASP יש לתת פקודה עבור כל תכונה של Cookie באופן הבא:

```
Response.cookies("gunga")=12
```

בפקודה זו, הגדרנו עוגיה בשם gunga ואחסנו בתוכה את הערך 12.

```
Response.cookies("gunga").expires=date
```

בשורה הזו הוגדר כי לעוגיה gunga, ייתנן תאריך תפוגה.

```
Response.cookies("gunga").path="/"
```

וכמובן, בשורה זו הוגדר Path עבור העוגיה.

# Count

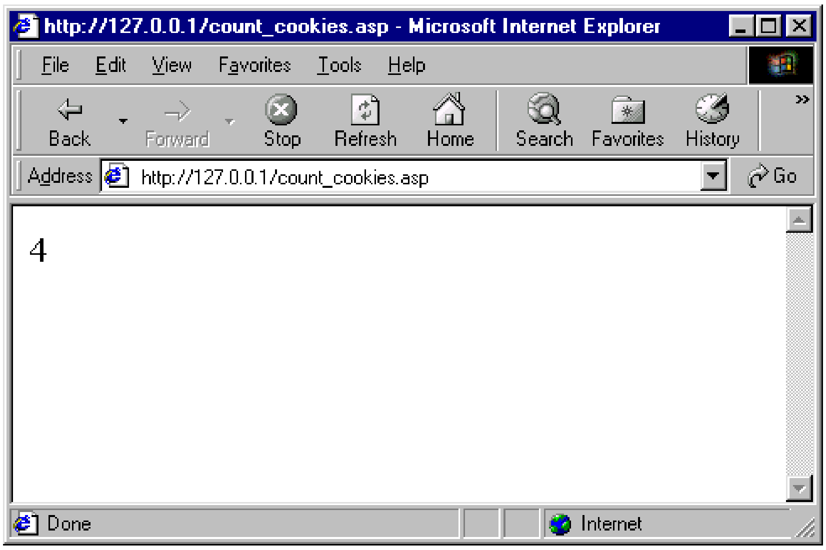
באמצעות התכונה count, ניתן לברר את מספר העוגיות שנרשמו אצל הגולש, על ידי היישום הנוכחי.

לדוגמה:

count_cookies.asp

```
<%
response.cookies("a")
response.cookies("b")
response.cookies("v")
response.cookies("f")
response.write "<font size=5>" & request.cookies.count
%>
```

ולהלן התוצאה:



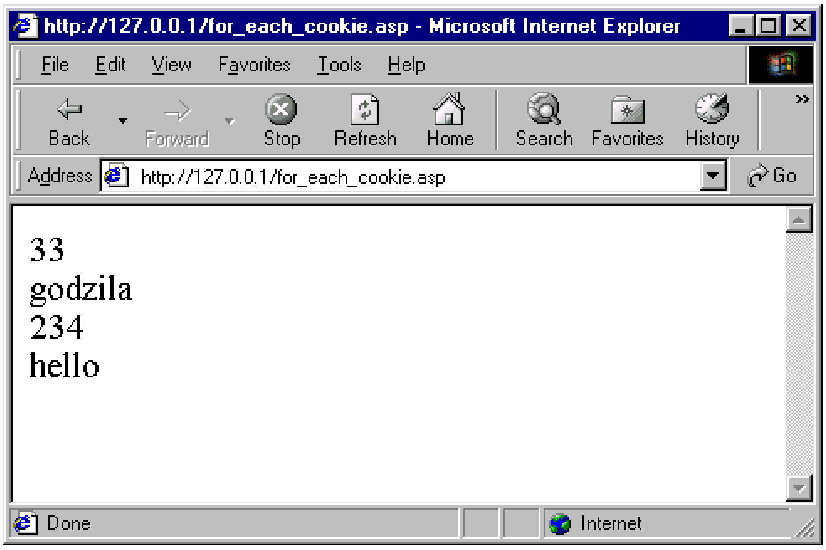
# For each

ASP מאפשר מעבר על העוגיות כמערך על ידי שימוש בלולאה for each באופן הבא :

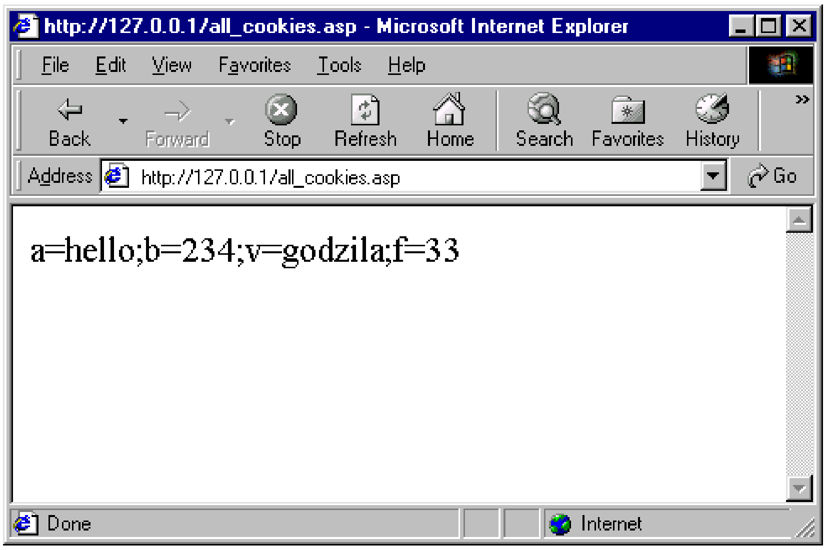
For_each_cookie.asp

```
<%
response.cookies("a")="hello"
response.cookies("b")=234
response.cookies("v")="godzila"
response.cookies("f")=33
for each item in request.cookies
response.write "<font size=5>" & request.cookies(item) & "<br>"
next
%>
```

כך תיראה התוצאה :




שימו לב, כי כל עוגיה הוצגה בשורה נפרדת מכיון שהגישה לכל עוגיה התבצעה כ- request.cookies(item) לעומת request.cookies בלבד, אשר היה מחזיר את התשובה הבאה :




כאמור, המסך הנ״ל נוצר כתוצאה משימוש בקוד הבא:

```
all_cookies.asp

<%
response.cookies("a")="hello"
response.cookies("b")=234
response.cookies("v")="godzila"
response.cookies("f")=33
response.write "<font size=5>" & request.cookies
%>
```

# מפתח

ניתן לאחסן מספר נתונים רב בתוך עוגיה על ידי שימוש במפתח (key). במקום לשמור מספר רב של עוגיות שונות, ניתן לתת לאותה עוגיה מפתחות שונים אשר יאפשרו לשלוף את המידע בקלות. לדוגמה, אם נרצה לאחסן פרטי משתמש כגון שם פרטי, שם משפחה וטלפון בנוחיות, נגדיר עוגיה אחת בשם user_details. בתוך עוגיה זו במבנה של מערך, נגדיר את שאר הנתונים בצורה הבאה:

```
response.cookies("user_details")("first_name")="moishe"
response.cookies("user_details ")("last_name")="blumenfeld"
response.cookies("user_details ")("phone_number")="02-985658"
```

כעת, ניתן לגשת לנתונים בצורה הפשוטה הבאה:

```
response.write request.cookies("user_details")
```

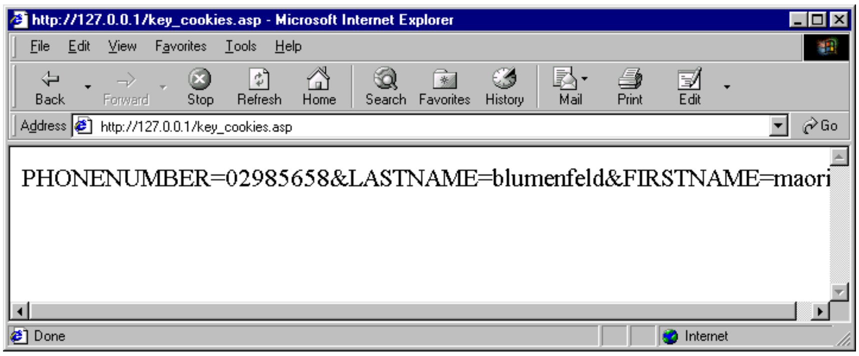
הקוד המלא ייראה כך :

key_cookies.asp

```
<%
response.cookies("user_details")("firstname")="maori"
response.cookies("user_details")("lastname")="blumenfeld"
response.cookies("user_details")("phonenumber")="02985658"

response.write "<font size=5>" &request.cookies("user_details")
%>
```

התוצאה תיראה כך :



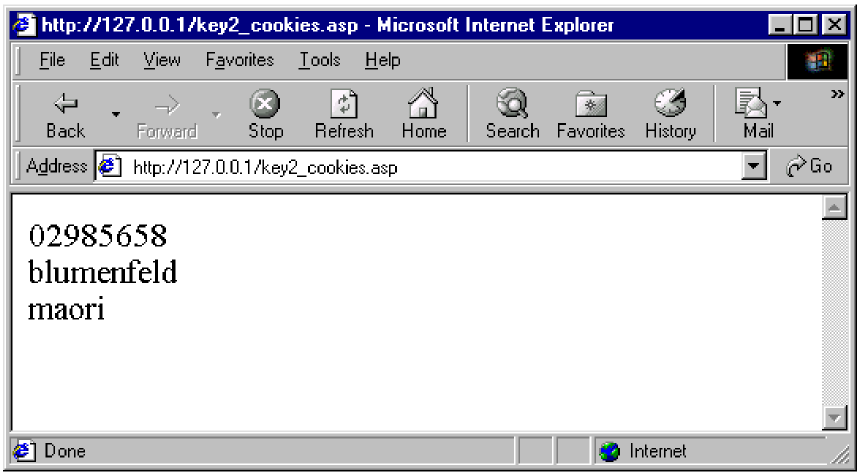

כפי שניתן לראות, קיבלנו את כל ערכי המפתחות כולל שמות המפתח. לא תמיד נרצה לקבל את המידע בצורת מחרוזת אחת, ולכן נוכל גם כאן להשתמש בלולאת for each באופן הבא :

key2_cookies.asp

```
<%
response.cookies("user_details")("firstname")="maori"
response.cookies("user_details")("lastname")="blumenfeld"
response.cookies("user_details")("phonenumber")="02985658"
for each item in request.cookies("user_details")
response.write "<font size=5>" &request.cookies("user_details")(item) & "<br>"

next
%>
```

אז תיראה התוצאה כך:




# טכניקות לייעול העבודה עם עוגיות

היתרון היחסי שיש לעוגיות, הוא העובדה שמכיון שהנתונים נשמרים אצל הלקוח, אין צורך במקום על גבי השרת לפרטיהם של מיליוני גולשים. עם זאת, הגישה לעוגיות לא תמיד נוחה ולעיתים קל יותר לגשת למשתנים מסוג session. לכן מומלץ לאחסן את הנתונים אצל הלקוח בפורמט של עוגיה אך להעביר את הנתונים למשתנים מסוג session עם התחברות הלקוח. אם נשתמש בדוגמה key_session.asp המאחסנת באופן יעיל את פרטי המשתמש ונגדיר כי:

```
response.cookies("user_details")("firstname")="maori"
response.cookies("user_details")("lastname")="blumenfeld"
response.cookies("user_details")("phonenumber")="02985658"
```

נוכל לבנות תסריט קצר המזין את הנתונים הללו לתוך משתנים מסוג session עם הגעתו של הגולש לאתר באופן הבא:

```
session("first_name")=request.cookies("user_details")(3)
session("last_name")=request.cookies("user_details")(2)
session("phone_number")=request.cookies("user_details")(1)
```

מרגע זה, נוכל להשתמש בנתוני Session בנוחיות בכל דף בתחום השימוש של המשתמש הספציפי. שימו לב כי הגישה למפתח העוגיה נעשה בסדר הפוך, כלומר, הגישה ל- first_name מתבצעת באמצעות מספר 3 בעוד שהגישה ל- phone_number מתבצעת באמצעות המספר 1.

# פרק 9

# מסדי נתונים

פרק זה הוא פרק עיוני אך חיוני. מכיון שרוב יישומי האינטרנט/אינטראנט כיום מבוססים מסדי נתונים, יש להבין את תפיסת מסד הנתונים ולהכיר את שפת הגישה אליהם. קורא המיודע עם שפת SQL רשאי לדלג לפרק הבא.

כל ארגון צובר נתונים. בין אם זהו בנק הצובר מידע על לקוחותיו, מצב חשבונותיהם והפעילות אותה הם מבצעים, ובין אם זוהי חנות קטנה אשר מאחסנת את נתוני המכירה והעבודה מול הספקים השונים. הצורך באחסון אותם נתונים הוא צורך אמיתי וחשוב, המוגבל על ידי הצורך לשלוף את אותם נתונים במהירות כדי להשתמש בהם בזמן אמת.

למרות עוצמתו הרבה של המחשב המודרני, עדיין קיימת מגבלת זמן על עיבוד נתונים רבים.

אחת הדוגמאות הקלאסיות מדברת על מסד נתונים המאחסן את פרטיהם האישיים של כל תושבי מדינת ישראל. כעת מבקשים ממסד הנתונים לשלוף את כל התושבים אשר שמם הפרטי הוא משה, ועיר מגוריהם היא ראשון לציון. על המחשב לעבור על 6 מיליון רשומות ובכל רשומה לבצע בדיקה ראשונית על שמו הפרטי של התושב ובמקרה של התאמה למשה, לבצע בדיקה נוספת על עיר המגורים. לכל הדעות פעולה ארוכה ומייגעת.

כעת, מחלקים את המידע במסד הנתונים לטבלאות. כל טבלה מכילה תושבי עיר מסוימת בארץ. לבקשת אחזור מידע דומה, יוכל המחשב לעבור רק על הטבלה המאחסנת את תושבי ראשון לציון.

התוצאה, תהליך בדיקה בודדת (בדיקת השם הפרטי) על כ- 350,000 רשומות בלבד.

ללא ספק, חלוקת המידע לטבלאות ספציפיות מייעלת את תהליך אחזור המידע בצורה משמעותית, אולם, יש דרכים רבות לייעל את תהליכי האחסון והאחזור. לדוגמה, רוב השמות הפרטיים של התושבים יחזרו על עצמם ובכך יוסיפו לזיכרון הדרוש לאחסון המידע. ניתן ליצור טבלה המכילה את כל השמות, אשר ממילא נכפה על מסד הנתונים להכיל, ולהעניק לכל שם מספר. בטבלת התושבים יופיע מספר זה במקום השם. בדרך כלל תופס המספר הרבה פחות זיכרון ממחרוזת השם, ולכן השמות עצמם יופיעו פעם אחת כל אחד, וכך נחסך מקום רב בזיכרון.

במקרים כאלה יש לבצע שיקול של חסכון במקום מול ביצועים, שכן על כל מספר בטבלת התושבים להיבדק מול טבלת השמות.

התהליך בו נבנה מסד הנתונים בצורה היעילה ביותר לאחסון ואחזור נקרא **'נירמול'** (גזור מהמונח Normalization) ודורש התמחות בתחום זה. עם זאת, לבניית יישומים פשוטים המכילים מספר טבלאות וכמות לא גדולה של מידע, אין צורך בהבנה נרחבת, כי אם בכמה כללי בסיס.

# SQL

יש סוגים רבים של מסדי נתונים המאחסנים מידע בדרכים שונות. מסדי הנתונים אליהם נתייחס בספר זה הם **מסדי נתונים יחסיים** (relational database) המכילים את המידע בטבלאות.

קיימים בשוק מסדי נתונים שנכתבו בשפות שונות, ולכן התעורר צורך לשפה סטנדרטית המאפשרת אחסון ושליפת מידע מכל מסד נתונים. שפה זו היא SQL (Structured Query Language).

שפה פשוטה זו מתחלקת לפקודות שליפה, הזנה ועדכון. נתחיל בדוגמה פשוטה.

## שליפת נתונים מטבלה

טבלת Phone_book מאחסנת ספר טלפונים קצר:

| **Phone_book** | | |
|---|---|---|
| Phone_number | Last_name | First_name |
| 321321 | Ben-Yaakov | Mooki |
| 123456 | Hurikan | Uri |
| 456578 | Hutch | Starski |

אם נרצה לשלוף את מספר הטלפון של מוקי בן-יעקב, נכתוב משפט SQL אשר יגדיר:

בחר את **שדה מספר הטלפון** מטבלת **Phone_book** היכן **ששדה השם הפרטי** שווה למוקי

נתקדם עוד צעד לכיוון שפת SQL ונרשום:

בחר את **Phone_number** מטבלת **Phone_book** היכן ש- **First_name**=Mooki

כעת נבחן את משפט SQL:

select **Phone_number** from **Phone_book** where **First_name**='Mooki'

משפט זה יחזיר את התשובה: 321321

אם נרצה לשלוף גם את מספר הטלפון וגם את שם המשפחה של מוקי, נכתוב:

select Pone_number,Last_name from Phone_book where First_name='Mooki'

משפט זה יחזיר את התשובה: 321321 Ben-Yaakov

כמו כן, נוכל לשלוף את כל שורת הנתונים על ידי שימוש בכוכבית (*):

select * from Phone_book where First_name='Mooki'

משפט זה יחזיר את התשובה: Mooki Ben-Yaakov 321321

שליפה אף יכולה להתבצע על פי שני מאפיינים, כגון שם פרטי ושם משפחה:

select * from Phone_book where First_name='Mooki' and Last_name='Ben-Yaakov'

# יצירת טבלה

התחביר ליצירת טבלה פשוט אף הוא: צור טבלה בשם Bad_things וקבע את העמודות הבאות (שם מסוג מחרוזת, דירוג מסוג מספר).

משפט SQL ייראה כך:

create table Bad_things (Name varchar, Rating number)

ניתן להבחין כי עמודת Name היא מסוג Varchar. בשל ההבדלים בין מסדי הנתונים, יש לדעת על איזה מסד נתונים אנו עובדים, למרות העובדה שניתן לפנות לכולם ב-SQL, ישנם שינויים קטנים כגון הגדרת סוגי השדות. Varchar מסמל שדה מסוג טקסט במסד נתונים מסוג Access. במסד נתונים מסוג Oracle ניאלץ להשתמש במונח Varchar2.

אם כן, זוהי הטבלה שיצרנו:

| **Bad_things** | |
|---|---|
| Rating | Name |

כמובן, עדיין לא קיימים נתונים בטבלה.

# הזנת נתונים לטבלה

משפט SQL להזנת נתונים לטבלה ייראה כך :

הזן לטבלת Bad_things לשדות (Name,Rating) את הערכים (Murder,10).

וכך ייראה התחביר האמיתי :

```
insert into Bad_things(Name,Rating) values('Murder',10)
```

התוצאה תתבטא כך :

| **Bad_things** | |
|---|---|
| Name | Rating |
| Murder | 10 |

ניתן לוותר על שמות השדות אם ההזנה תואמת את מספר וסוגי השדות, לדוגמה :

```
insert into Bad_things values('Theft',4)
```

הפעולה תתבצע ללא הבדל :

| **Bad_things** | |
|---|---|
| Name | Rating |
| Murder | 10 |
| Theft | 4 |

# עדכון טבלה

לעיתים עלינו לעדכן מידע הרשום בטבלה. התחביר לעדכון ייראה כך :

```
update Bad_things set Name='Drug Use' where Name='Theft'
```

התוצאה תתבטא כך :

| **Bad_things** | |
|---|---|
| Name | Rating |
| Murder | 10 |
| Drug Use | 4 |

## ביטול טבלה

לביטולה של טבלה השתמשו בתחביר הפשוט:

```
drop table Bad_things
```

שפת SQL היא שפה פשוטה בבסיסה, אולם חשוב לציין כי בבניית יישומים אשר אינם פשוטים לחלוטין ומשלבים קשרים בין טבלאות ובדיקות מסובכות, יש להכיר טוב את השפה ולהתנסות בה. המלצת מחברי ספר זה היא, ללמוד על בוריה את השפה משום שיישומיה רבים כמעט בכל סביבת פיתוח, לרבות יישומי האינטרנט. ניתן למצוא reference בסיסי לשפת SQL בפרק 18.

## מחיקת רשומות מטבלה

מחיקת רשומות מטבלה מתבצעת באמצעות הפקודה delete.

אם נרצה למחוק את השורה הראשונה בטבלה שיצרנו, נרשום

מחק מטבלה Bad_things היכן שעמודת Name שווה ל-Murder

כך שהתחביר יהיה:

```
Delete from Bad_things where Name='Murder'
```

להעמקה בנושא SQL אנו ממליצים על הספר שיצא בהוצאת הוד-עמי:

**בסיסי נתונים טבלאיים ושפת SQL - עקרונות ועיצוב**

ראה פרטים בקטלוג בסוף הספר ובתקליטור המצורף.

# פרק 10

# מודל ADO

בפרק זה נלמד כיצד לשמור מידע דרך דפי ASP, וכן, כיצד להציג דפים דינמיים מנתונים השמורים על השרת.

## ODBC

כדי להבין כיצד ניגשים מ- ASP למסדי נתונים, בחרנו בספר זה להסביר תחילה את מהות ODBC.

מכיון שמסדי נתונים שונים מתקשרים בשפות שונות, יש צורך בגורם מתווך אשר יידע להסב את שפת SQL המשותפת לכולם, לשפה הייחודית של כל מסד נתונים. גורם זה נקרא **Driver**.

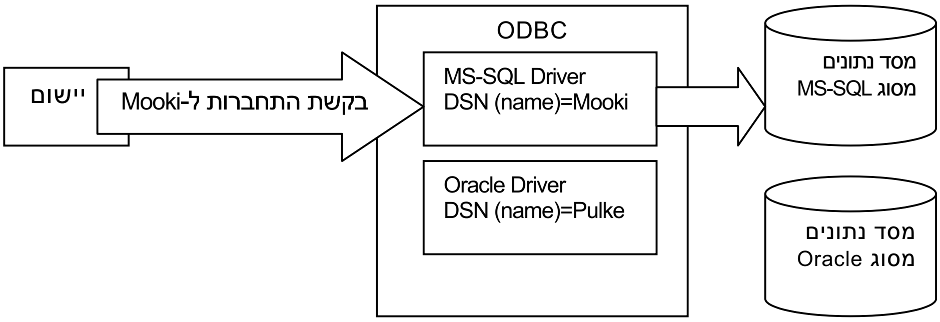
Driver מאפשר להתחבר אל מסד הנתונים ולהעביר לו את משפטי SQL, כך שניתן להתעלם מדרך החיבור, המשתנה ממסד נתונים אחד למשנהו, ומשפת השאילתות, ולבצע חיבור פשוט בשפת SQL תוך התעלמות (חלקית לפחות) מסוג מסד הנתונים.

ODBC בבסיסו הנו קופסת Drivers. ניתן להתקין בו Drivers שונים וליצור חיבורים לוגיים למסד נתונים.

לדוגמה, אם נבחר להתחבר למסד נתונים של Microsoft הנקרא SQL, נוכל לבקש מ-ODBC ליצור חיבור למסד הנתונים בהתבסס על SQL Driver המותקן בו. לחיבור נעניק שם פשוט.

בכל פעם שנרצה בחיבור למסד נתונים, נבקש מ-ODBC את השם הפשוט שהענקנו לחיבור. ODBC ידאג לכל השאר. השם שנעניק לחיבור נקרא **DSN** (Data Source Name).

להלן תרשים המדגים את פעולת **ODBC**:



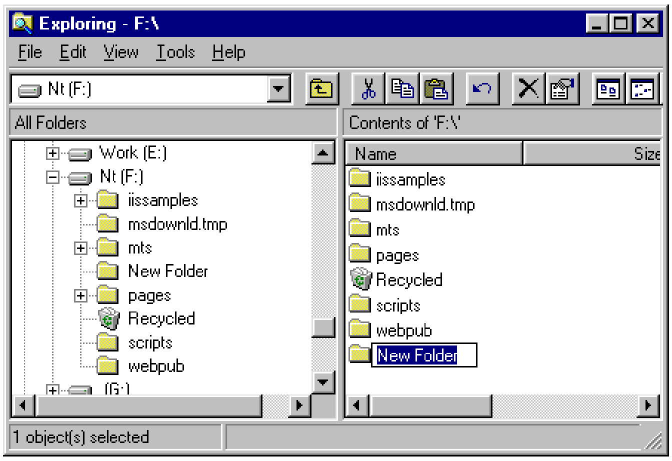

הסיכוי שכל קוראי הספר משתמשים באותו מסד נתונים הוא קטן, ולכן נתמקד בספר זה בשני מסדי נתונים. האחד, ובו נדון בפרק זה, הינו קובץ טקסט, והשני הוא Microsoft Access.

# קבצי טקסט כמסד נתונים

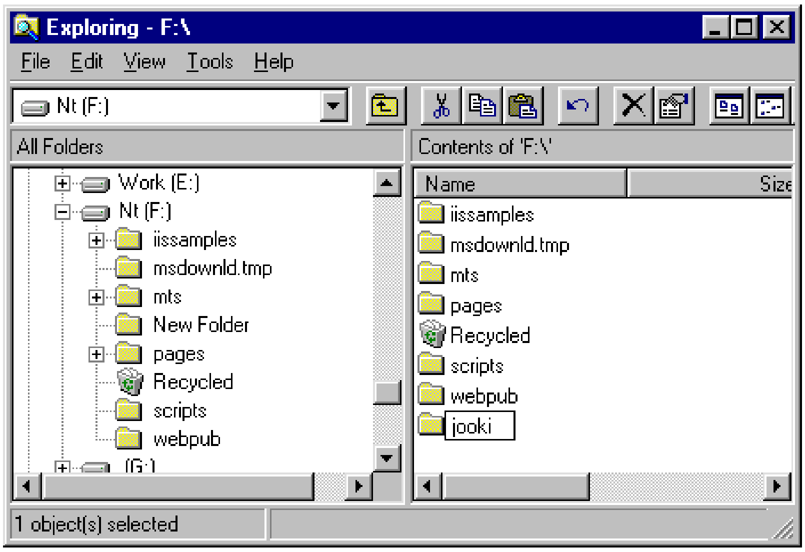
גם ללא תוכנת מסד נתונים, ניתן לאחסן ולאחזר מידע בצורה יעילה. זאת נבצע על גבי קבצי טקסט. Microsoft Text Driver יסייע בידינו להתייחס לקבצי הטקסט כאל טבלאות לכל דבר. מכיון שתפקידי ODBC הוא להפוך את הגישה לנתונים לשקופה למפתח, יעלה בידינו לבנות קוד יישומי מלא.

## תהליך הגדרת חיבור למסד נתונים מבוסס קבצי טקסט

נפתחת תיקיה (Folder) מיוחדת עבור קבצי הטקסט אשר ישמשו אותנו כמסד נתונים.



נקרא לתיקיה בשם jooki.




לאחר מכן נלחץ על **התחל** (Start) 


נבחר בתפריט **הגדרות** (Settings) ← **לוח בקרה** (Control Panel).

לאחר מכן, נאתר ונלחץ לחיצה כפולה (Double Click) על סמל **ODBC**




(ייתכן שבמחשבים מסוימים יצוין רק ODBC)

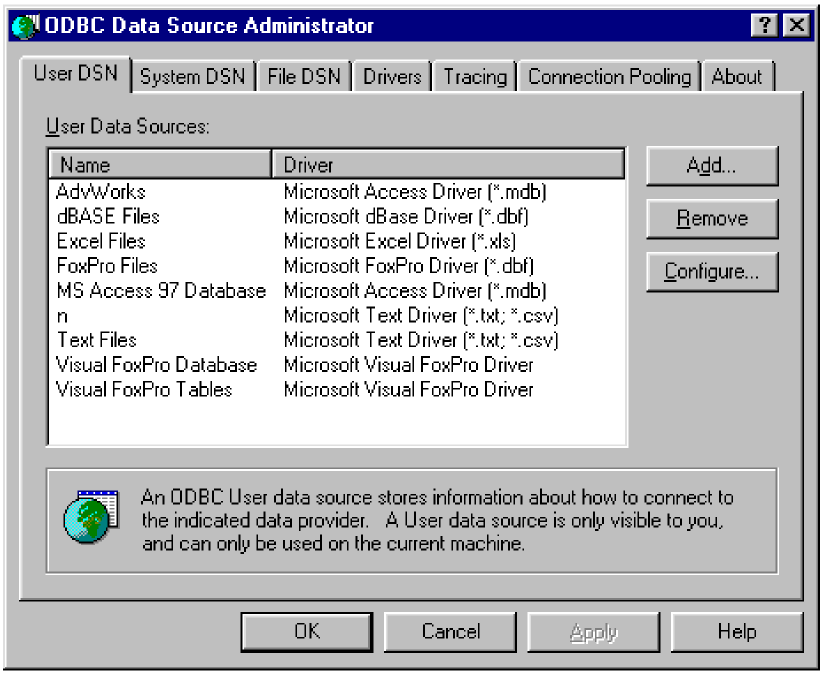
אם אין במחשב את הסמל כלל, יש להיכנס לאתר מיקרוסופט, בכתובת:

www.microsoft.com/data/mdac2.htm

ולהוריד את הגירסה האחרונה של ODBC.

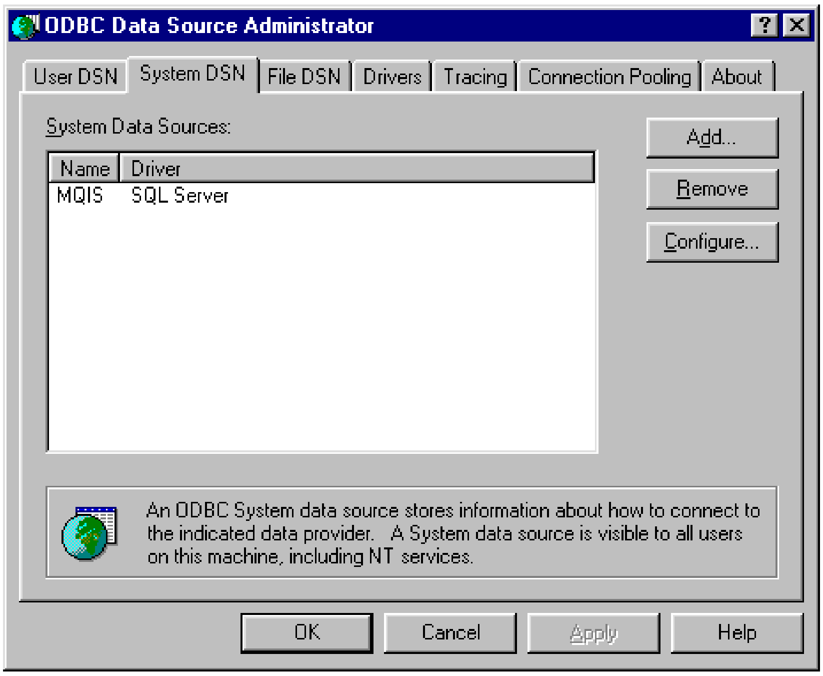
בכל מקרה מומלץ להוריד את הגירסה העדכנית.

זהו המסך שייפתח בפנינו :



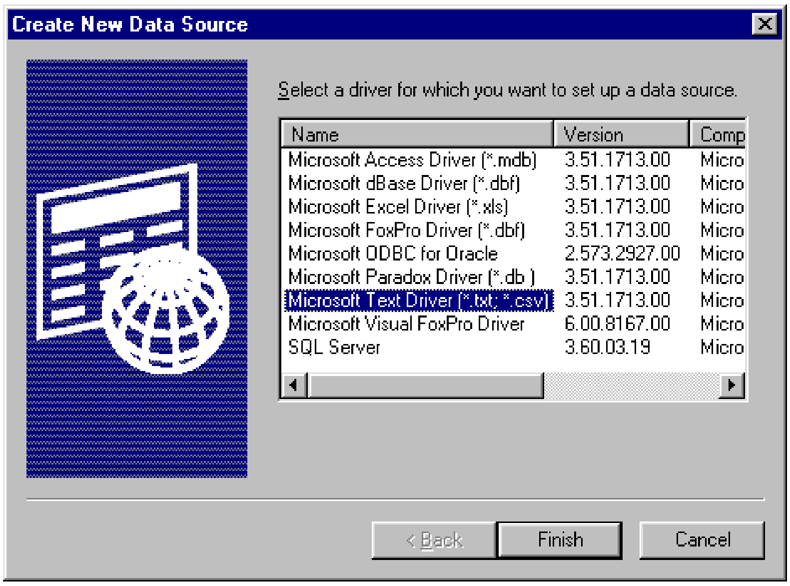

במסך זה ניתן לקבוע קשרים למסדי נתונים ברמת User (לפי שם לשונית הכרטיסיה - user DSN). כלומר, רק המשתמש שמגדיר את הקישור למסד הנתונים יוכל להשתמש בו.

נבחר בלשונית System DSN (כזכור, Data Source Name הוא השם הלוגי לחיבור הספציפי) ונקבל את המסך הבא:



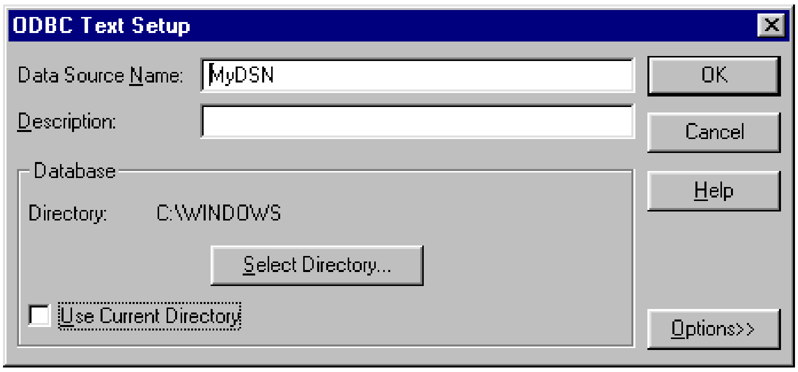

מסך זה מאפשר הגדרת חיבור למסד נתונים, המאפשר לכל משתמשי המערכת להתחבר דרכו.

נלחץ על לחצן **הוסף** (**Add**) ונקבל את המסך הבא:

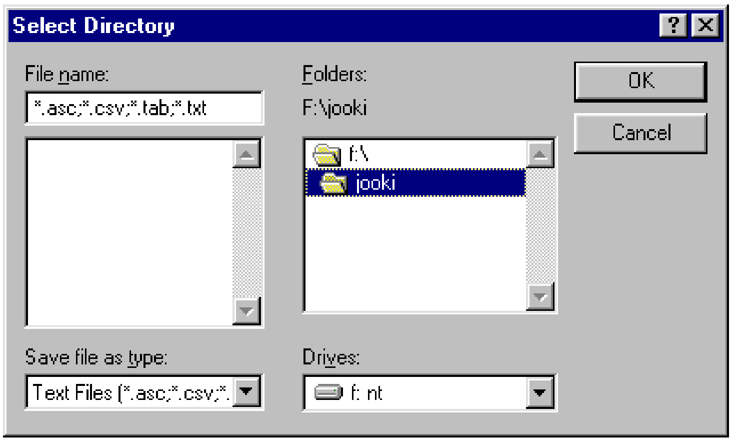


מסך זה מראה את כל ה- Drivers המותקנים ב- ODBC. נבחר את Microsoft Text Driver ונלחץ על **סיים** (**Finish**).

זהו המסך שנקבל:




במסך זה נקבע את שם החיבור כ- MyDSN בשדה הראשון. זכרו כי כעת נקרא החיבור שלכם למסד הנתונים – MyDSN. לאחר מכן נבטל את הסימון בתיבת הסימון התחתונה (Use Current Directory). לאחר שנבטל את הסימון, נוכל ללחוץ על הלחצן **Select Directory** ונקבל את המסך הבא:




נבחר את תיקיית jooki החדשה ונלחץ **OK**.

לאחר מכן נלחץ **OK** פעמיים (בשני המסכים הקודמים) וכעת אנו מוכנים להתחיל בעבודה.

# סיכום ביניים והבהרה

לאחר שביקשנו מ-ODBC להעניק שם לוגי (MyDSN) לחיבור לקבצי הטקסט, לא מעניין אותנו אם המידע נשמר במסד נתונים של Oracle, מיקרוסופט או בקבצי טקסט (שימו לב כי עדיין לא הבנו מה הכוונה בשמירת מידע בקבצי טקסט!). אנו מאחסנים ושולפים מידע מ-MyDSN. זוהי גדולתו ופשטותו של ODBC המאפשר לנו לבצע משפטי SQL פשוטים בלי להתחשב באופן בו הנתונים נשמרים בסופו של דבר. למעשה, מרגע שהוגדר DSN, יכול המפתח לשכוח מאופן אחסון הנתונים.

# האובייקטים של ADO

אם כן, כל שנותר כעת הוא לשלוח משפטי SQL אל MyDSN כגון יצירת טבלה, שליפת מידע וכו׳. כדי לפנות ל-MyDSN, יש לבצע קריאה מסודרת מ-ASP ל- ODBC. כאן נכנסים לעזרתנו האובייקטים של ADO.

ראשי התיבות של ADO הם **A**ctive **D**ata **O**bject, אך ייתכן שתיתקלו גם ב- **A**ctiveX **D**ata **O**bject.

הרעיון העומד מאחורי ADO הוא לספק מספר אובייקטים מוכנים מראש אשר יאפשרו גישה קלה למסדי נתונים דרך מתווכים כגון ODBC (ללא האובייקטים המוכנים מראש של ADO, היינו נאלצים לכתוב שורות קוד רבות ומסובכות כדי להתחבר ל- ODBC). האובייקטים בהם נדון בספר זה הם :

**Connection** – אובייקט המאפשר התחברות קלה אל DSN שהוגדר ב- ODBC.

**Recordset** – אובייקט שישרת אותנו בשליפת מידע ממסד הנתונים.

## אובייקט Connection

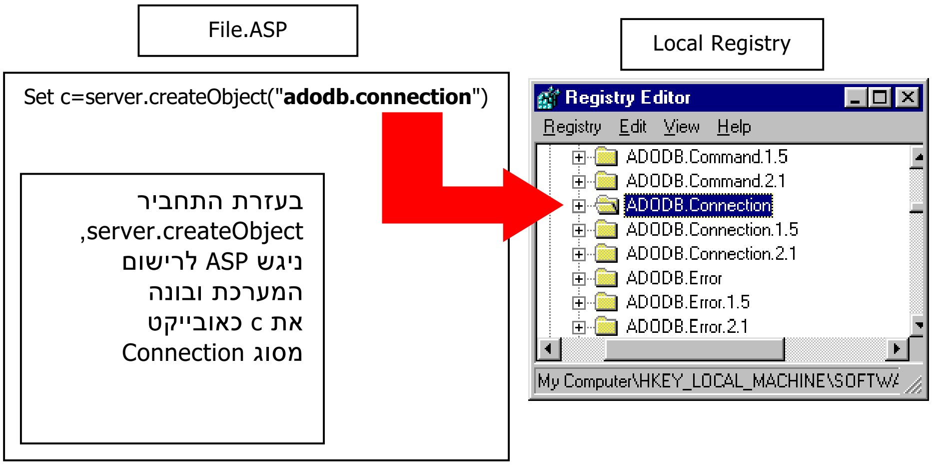
אובייקט זה יבסס עבורנו קשר למסד הנתונים. התחביר ליצירת אובייקט ADO אינו פשוט במבט ראשון, אך הלוגיקה מאחוריו היא פשוטה. התחביר הוא :

```
set c=server.createobject("adodb.connection")
```

המילה set מאפשרת לאתחל את c כאובייקט ולא כמשתנה רגיל.

השימוש באובייקט ובשיטה server.createobject מורים למנוע ASP לגשת לאובייקטים הרשומים במערכת (אובייקטים אלה מותקנים עם התקנת השרת) ולייצר את האובייקט c על פי התבנית המוכנה שם. במקרה זה, אובייקט Connection.

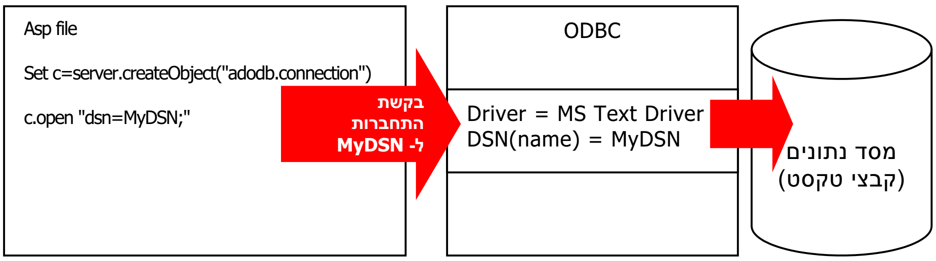
תחביר זה מאתחל את המשתנה c כאובייקט מסוג Connection.




כעת, יש להגדיר לאובייקט Connection (c) לאיזה DSN עליו להתחבר. זאת נבצע על ידי השיטה open של אובייקט Connection.

c.open "dsn=MyDSN;"

התרשים הבא ממחיש כיצד פונה ASP למסד הנתונים דרך ODBC.




בשלב זה, לאחר שקבענו כי c הינו אובייקט מסוג Connection וכי עליו להתחבר ל- DSN שהוגדר ב- ODBC, ניתן לבצע פקודות SQL על ידי שיטת Execute של אובייקט Connection.

c.execute "create table first_table (Name char, Tel char)"

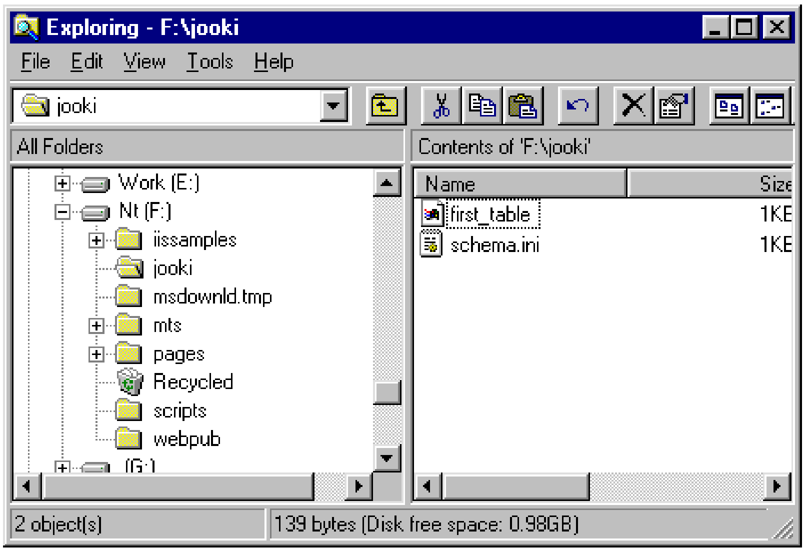
משפט SQL זה ייצר טבלה בשם first_table ובה שתי עמודות. עמודת name מסוג Char (מלל), ועמודת Tel מסוג Char אף היא.

להלן הקוד המלא:

table_creation.asp

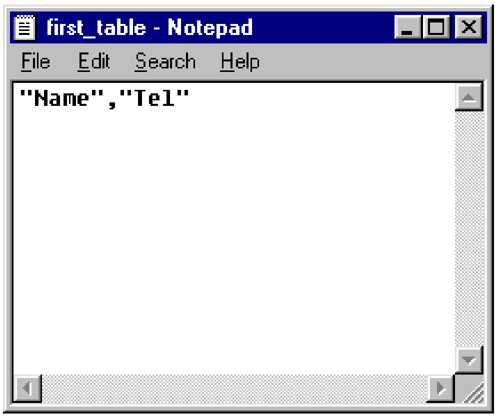
```
<%
set c=server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
c.execute "create table first_table (Name char, Tel char)"
%>
Your table has been created
```

הריצו קוד זה. לאחר מכן הציצו בתיקיית jooki החדשה ותגלו כי נוצרו שם שני קבצים: קובץ בשם first_table, וקובץ בשם schema.ini.




אם תפתחו את הקובץ first_table בעזרת פנקס הרשימות (notepad), תגלו כי זו למעשה הטבלה שיצרתם. אל תשנו בקובץ זה דבר.

כך ייראה הקובץ first_table :




כפי שבוודאי ניחשתם, הנתונים בטבלה מוקפים בגרשיים ומופרדים בפסיק. ניתן לשרטט את הטבלה החדשה שיצרנו כך :

| First_table | |
|---|---|
| **Name** | **Tel** |
| | |
| | |

אם כן, יצרנו טבלה על השרת דרך קובץ ASP. כל שנותר הוא להזין מידע אל הטבלה.

## הזנת נתונים לטבלה

כעת, נכין קובץ ASP חדש אשר יזין נתונים אל הטבלה החדשה שיצרנו. ניצור אובייקט connection כפי שביצענו בתרגיל הקודם, על ידי שימוש ב- server.createobject.

```
set c=server.createobject("adodb.connection")
```

נגדיר שוב את DSN, אליו אנו מתחברים, על ידי השיטה open :

```
c.open "dsn=MyDSN;"
```

והפעם נבצע insert אל הטבלה. נבצע זאת מספר פעמים :

```
c.execute "insert into first_table values('mooki','1234')"
c.execute "insert into first_table values('joe','4321')"
c.execute "insert into first_table values('cabuto','6690')"
```

חידוש נוסף בקוד זה, הפעם נסגור את אובייקט connection על ידי שימוש בשיטה Close, כדי לא להכביד על מסד הנתונים. אמנם במקרה זה אין אנו מתחברים למסד נתונים, אך מומלץ לסגור את Connection עם סיום הפעולה כדי למנוע מצב בו משתמשים רבים משאירים Connections פתוחים וסותמים את מסד הנתונים.

```
c.close
```

אם נרצה לבטל לחלוטין את אובייקט Connection כדי לחסוך במשאבי מערכת, נשתמש בתחביר:

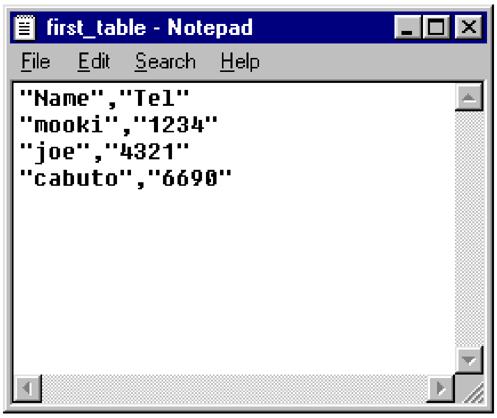
```
set c=nothing
```

להלן הקוד המלא:

table_insert.asp

```
<%
set c=server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
c.execute "insert into first_table values('mooki','1234')"
c.execute "insert into first_table values('joe','4321')"
c.execute "insert into first_table values('cabuto','6690')"
c.close
set c=nothing
%>
The information was submitted to the database
```

לאחר הרצת הקוד בהצלחה, נפתח שוב את הקובץ first_table ונראה כי המידע אכן נכנס לטבלה:



כמובן כי לצרכי הבהרה ניתן לשרטט טבלה זו כך :

| First_table | |
|---|---|
| **Name** | **Tel** |
| Mooki | 1234 |
| joe | 4321 |
| Cabuto | 6690 |

## תרגיל הכנת רשימת אורחים באתר

לפני שמתקדמים הלאה בספר, מומלץ לתרגל יצירת טבלאות בדרך זו מתוך דפי ASP שונים. לפיכך נבצע תרגיל בו נאפשר למבקרים באתר להירשם ברשימת האורחים. לצורך תרגיל זה נשתמש באותו DSN שיצרנו (MyDSN).

מטרת התרגיל היא לבקש את פרטיו האישיים של מבקר ולרשום אותם בטבלה מבוססת קבצי טקסט.

נתחיל במסמך ASP אשר יבנה את הטבלה. נבסס את c כאובייקט Connection :

```
set c=server.createobject("adodb.connection")
```

נחבר את אובייקט Connection שלנו ל- DSN המוגדר ב- ODBC, הלא הוא MyDSN :

```
c.open "dsn=MyDSN;"
```

וניצור את הטבלה guest_book המכילה שם פרטי, שם משפחה ודואר אלקטרוני. כל השדות הללו יהיו מסוג char, כלומר יכילו מחרוזת טקסט שאינה עולה על 255 תווים.

```
c.execute "create table guest_book(fname char,lname char,email char)"
```

נסגור את אובייקט Conection ונבטל אותו מזיכרון היישום :

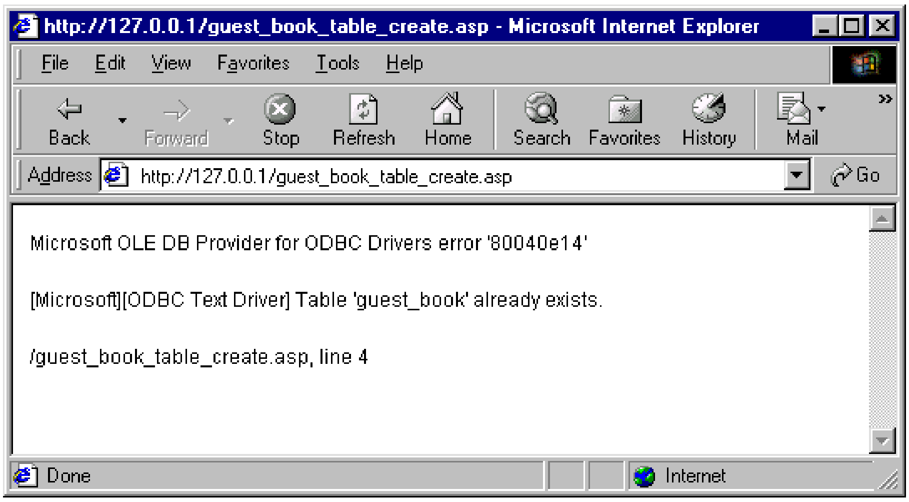
```
c.close
set c=nothing
```

לפניכם הקוד המלא, כולל מלל HTML אשר ייתן לנו אינדיקציה כי הקוד התבצע:

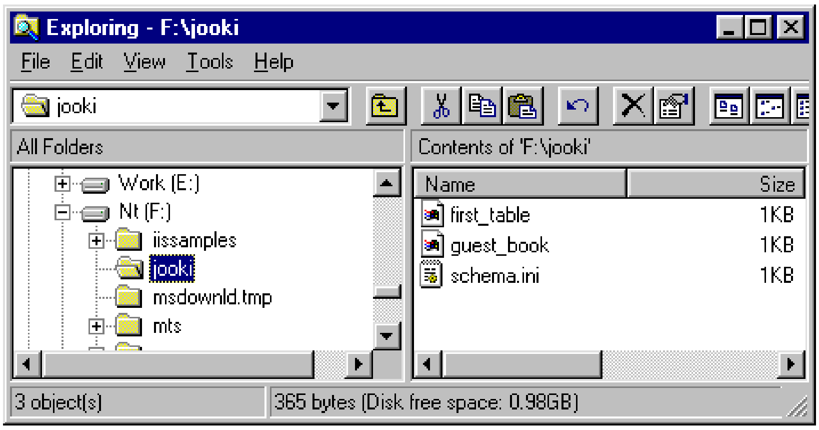
guest_book_table_create.asp

```
<%
set c=server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
c.execute "create table guest_book(fname char,lname char,email char)"
c.close
set c=nothing
%>
Guest_book table has been created successfully!
```

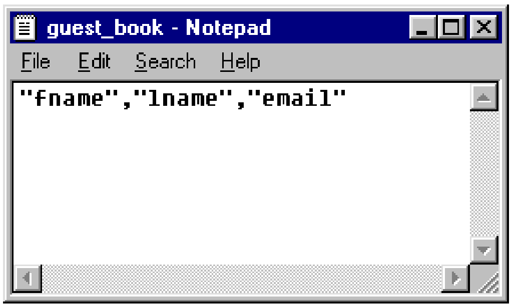
נריץ קוד זה פעם אחת. אם נריץ קוד זה יותר מפעם אחת עלולה להתעורר בעיה, מכיוון שהטבלה אותה מנסה קוד זה ליצור, כבר קיימת. זו הודעת השגיאה העלולה להופיע אם נריץ קוד זה יותר מפעם אחת:



לאחר הרצת הקובץ פעם אחת בלבד, נראה כי בתיקיית jooki נוצר הקובץ guest_book.




אם נפתח את הקובץ guest_book בעזרת פנקס הרשימות (notepad), נוכל לראות את מבנה הטבלה:



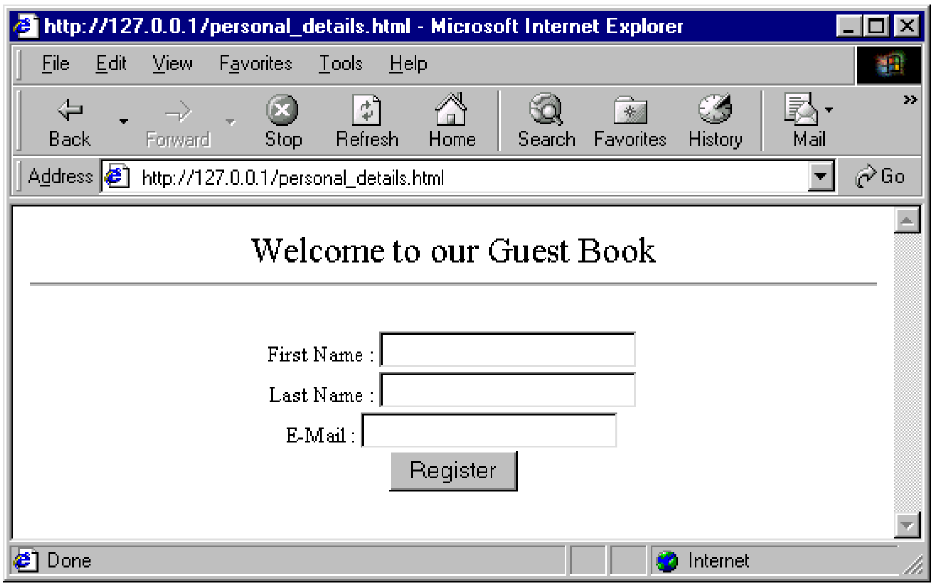

כעת, נבנה מסמך HTML המכיל את טופס קבלת הפרטים:

personal_details.html

```
<body>
<center>
<font size=5> Welcome to our Guest Book</font><hr>
<form action=personal_details_insert_into_guest_book.asp>
First Name : <input type=text name=fn><br>
Last Name : <input type=text name=ln><br>
E-Mail : <input type=text name=em><br>
<input type=submit value=Register>
</form>
```

חשוב לציין, כי כל יישום אינטרנט בנוי מתכנות בצד שרת, כגון ASP שאנו בונים, ותכנות בצד לקוח. כיון שספר זה אינו דן בתכנות בצד לקוח, אין בקוד זה שימוש ב-JavaScript, אך חובה על יישום שכזה לשלב בדיקות JavaScript בצד הלקוח המחייבות מילוי של כל השדות בטרם שליחה ובדיקת הזנת כתובת דואר אלקטרוני תקינה וכו'.

כך ייראה מסך HTML :




כעת עלינו לבנות מסך ASP, אשר יקבל את הנתונים מהטופס ויזין אותם אל הטבלה.

נבסס שוב אובייקט Connection ונגדיר DSN :

```
set c=server.createobject("adodb.connection")
c.open "dsn=DSN;"
```

כעת נבצע insert תוך שרשור הנתונים שהגיעו מהלקוח :

```
c.execute "insert into guest_book values('" & request.querystring("fn") & "','" &_
request.querystring("ln") & "','" & request.querystring("em") & "')"
```

שימו לב, כי השרשור אינו פשוט, כיון שיש לכתוב את הגרשיים הבודדים (') המחויבים בתחביר SQL ← ('first name','last name','email'), וכן את הגרשיים הכפולים (״) המחויבים בתחביר השרשור של ASP ← "insert into..." & request...

לאחר מכן נסגור ונבטל את אובייקט Connection :

```
c.close
set c=nothing
```

להלן הקוד המלא הכולל גם את מלל HTML שיופיע על המסך עם סיום הפעולה (שם הקובץ הוא כמובן השם המופיע ב- Action של הטופס ב-HTML):

personal_details_insert_into_guest_book.asp

```
<%
set c=server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
c.execute "insert into guest_book values('" & request.querystring("fn") & "','" &_
request.querystring("ln") & "','" & request.querystring("em") & "')"
c.close
set c=nothing
%>
You have been registered successfully!
<a href=personal_details.html>Back to form</a>
```

**הערה**:
כמובן שאין לפנות ישירות לקובץ personal_details_insert_into_guest_book.asp, כיון שאז לא יישלחו אליו הנתונים fn,ln,em ותתבצע הזנה ריקה לטבלה! ניתן לפתור בעיה זו על ידי בדיקת isempty פשוטה. כלומר, להתנות את ביצוע הקוד בתנאי:

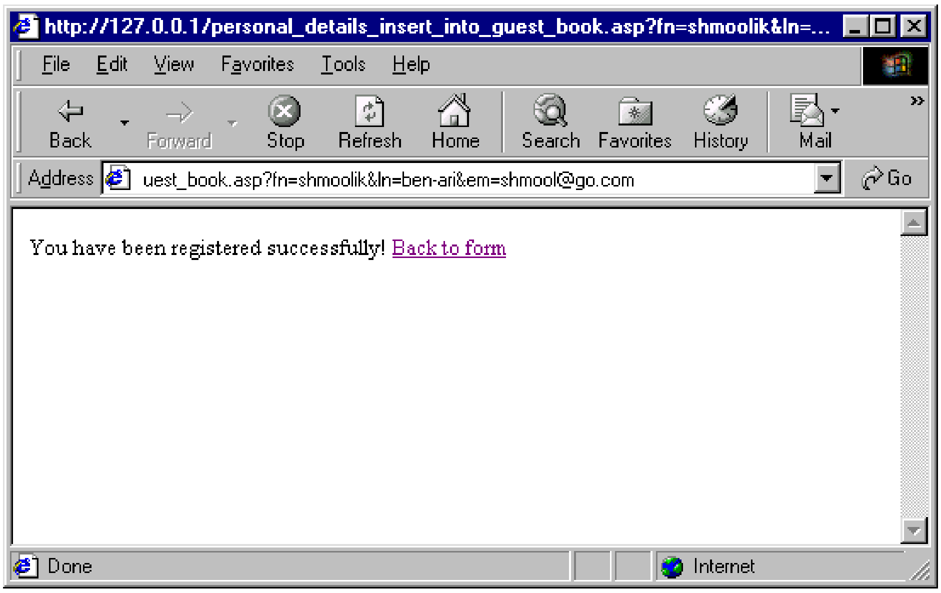
```
if isempty(request.querystring("fn")) then
```

ולא לשכוח end if בסוף!

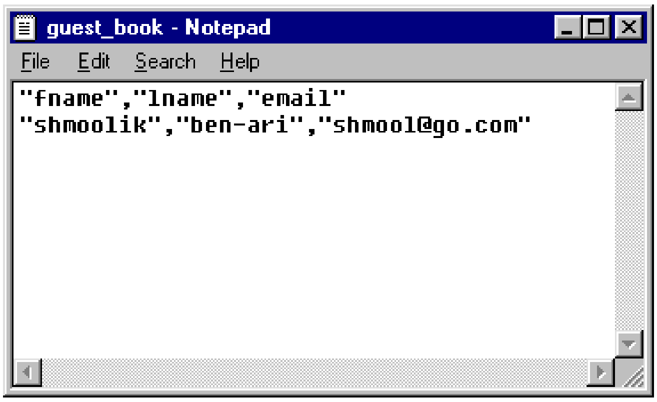
הבה נעקוב אחר צילומי המסך של היישום. נתחיל במסך HTML:



לאחר מילוי הנתונים נקבל:




כל שנותר, הוא לפתוח שוב את הקובץ guest_book מתיקיית jooki ולראות כי הנתונים הוזנו:



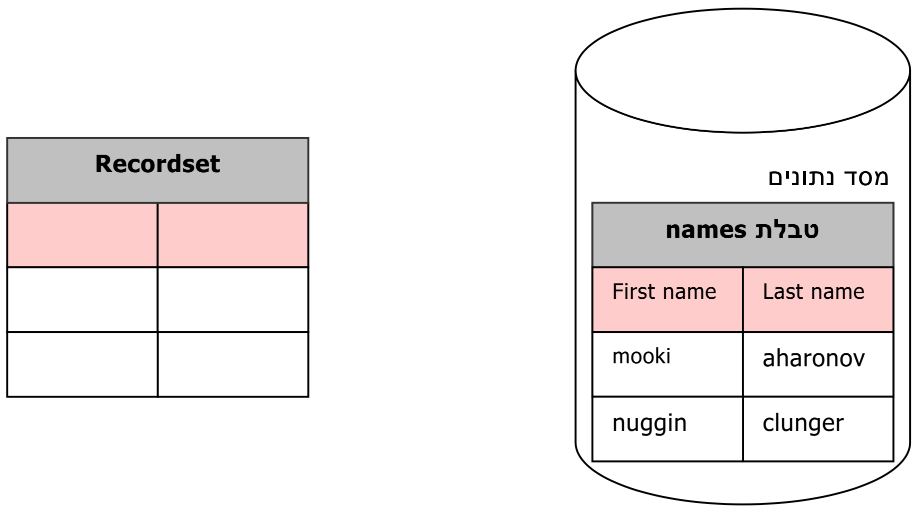
# שליפת נתונים והצגתם ב- Web

כדי לשלוף נתונים ממסד נתונים ולהציגם, יש להכיר אובייקט נוסף של ADO הנקרא Recordset.

## Recordset

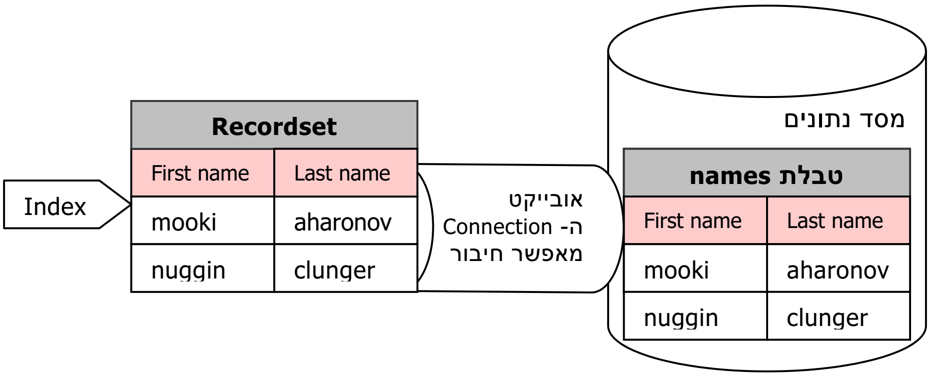
לצרכי הבנה, אם נשלוף נתון אחד ממסד נתונים, כגון שם פרטי של אדם, לא תהיה בעיה לאחסן אותו נתון בתוך משתנה רגיל. לעומת זאת, אם נרצה לשלוף את כל הנתונים מטבלה, לא נוכל לאחסן את המידע במשתנה רגיל. אנו זקוקים אם כך, למשתנה שיכול להכיל מידע טבלאי. זהו אובייקט Recordset. אובייקט Recordset הוא משתנה טבלאי המעניק לנו כלים נוחים לגשת למידע. הבה נדמה תהליך שליפת מידע מטבלה במסד הנתונים, לתוך אובייקט Recordset.

### שלב א




בשלב זה, לאחר שנוצר אובייקט Recordset, הוא ממתין ריק. כדי לאפשר ל-Recordset לשלוף את המידע מטבלת מסד הנתונים, יש להגדיר לו כיצד להתחבר דרך אובייקט Connection. לאחר ההתחברות, שולף Recordset את המידע מהטבלה ומאחסן אותה אצלו, כולל שמות העמודות.

## שלב ב




בשלב זה, לוקח אובייקט Recordset את הנתונים מטבלת names ומאחסן אותם בצורת טבלה. כיון שאובייקט Recordset נמצא במסמך ASP, ניתן כעת לגשת למידע דרך קוד ASP.

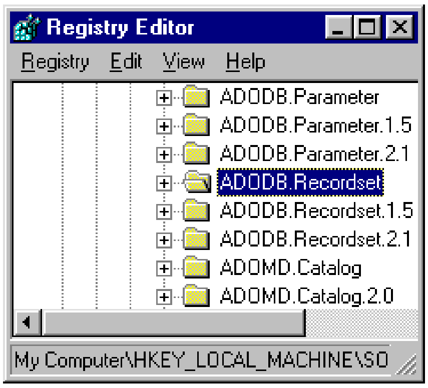
שים לב שבתרשים נוסף אלמנט בשם Index. אלמנט זה הוא אלמנט לוגי המצביע על שורה בטבלה. כפי שמופיע בתרשים, Index מתחיל בשורה הראשונה, כך שאם נבקש מ-Recordset את עמודת First Name, נקבל את mooki. אם נרצה לקבל מידע משורות אחרות, ניאלץ להזיז את Index שורה אחת קדימה ולבקש שוב את עמודת First name, ואז נקבל את nuggin.

# תחביר Recordset

יצירת אובייקט Recordset זהה ליצירת אובייקט Connection :

```
set r=server.createobject("adodb.recordset")
```

השימוש בתחביר server.createobject, מורה למנוע ASP לגשת לרישום המערכת ולאתחל אובייקט על פי אובייקט Recordset הרשום שם :



השלב הבא הוא הגדרת אובייקט Connection אשר דרכו ימשוך אובייקט Recordest את המידע ממסד הנתונים. אם אתחלנו אובייקט Connection בשם c, נשתמש בתכונה activeconnection של אובייקט Recordset כדי להגדיר מול איזה Connection נעבוד:

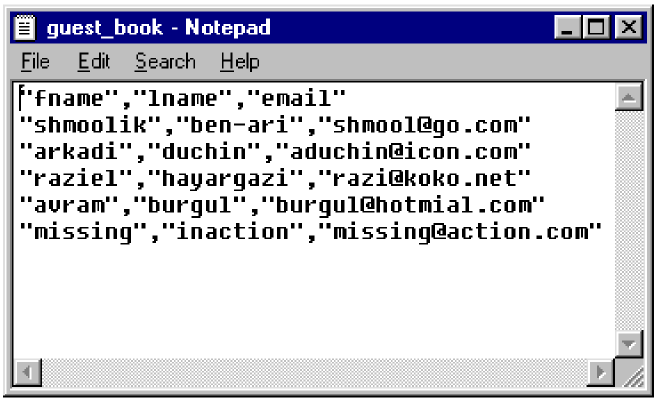
```
r.activeconnection=c
```

השלב הבא, יהיה שליפת המידע מטבלת guest_book אשר הגדרנו בתרגילים הקודמים. שליפת המידע מתבצעת באמצעות התכונה open של אובייקט Recordset:

```
r.open "select * from guest_book"
```

כעת, מכיל אובייקט Recordset שלנו (r) את כל המידע מהטבלה.

**הערה**: בטרם נמשיך, יש להיכנס ל- 127.0.0.1/personal_details.html ולהזין עוד מספר שמות לטבלה כך שנקבל מספר שורות כגון:




כעת, נחלץ את המידע מהשורה הראשונה ונציג אותה בדף Web.

זהו קטע הקוד שכתבנו עד כה:

```
'this is the connection object creation
set c=server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
'this is the Recordset object creation
set r=server.createobject("adodb.recordset")
r.activeconnection=c
r.open "select * from guest_book"
```

כיון ש-Index נמצא בשורה הראשונה, כל שעלינו לעשות הוא לבקש מאובייקט Recordset להציג את ערך העמודות fname, lname, email. גישה לעמודה ב- Recordset מתבצעת באמצעות השיטה fields באופן הבא:

```
r.fields("fname")
```

אם כן, נציג למשתמש את התוצאה:

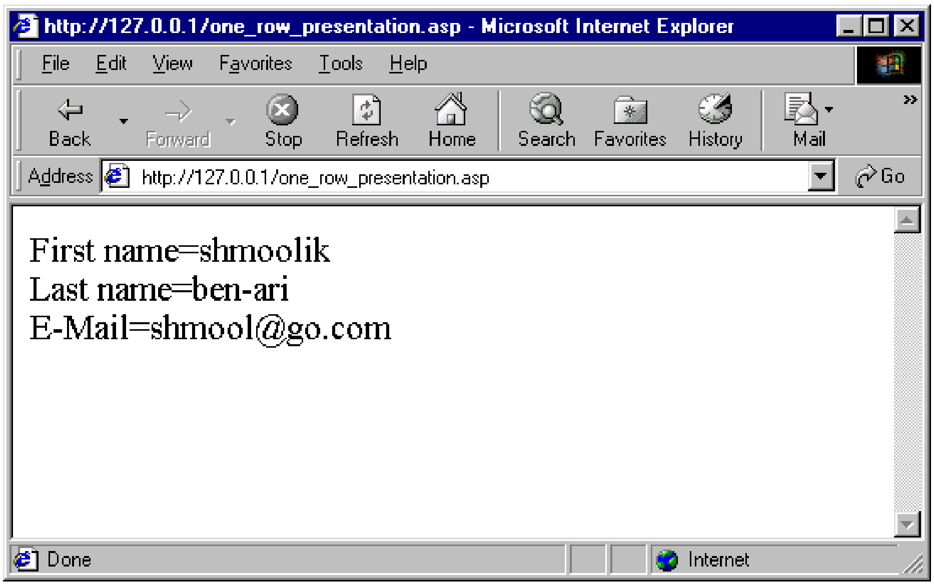
```
response.write "First name=" & r.fields("fname") & "<br>"
response.write "Last name=" & r.fields("lname") & "<br>"
response.write "E-Mail=" & r.fields("email") & "<br>"
```

הקוד המלא ייראה כך:

```
one_row_presentation.asp

<%
set c=server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
set r=server.createobject("adodb.recordset")
r.activeconnection=c
r.open "select * from guest_book"
response.write "<font size=5>"
response.write "First name=" & r.fields("fname") & "<br>"
response.write "Last name=" & r.fields("lname") & "<br>"
response.write "E-Mail=" & r.fields("email") & "<br>"
%>
```

כך תיראה התוצאה:




מזל טוב! הצגנו מידע ממסד נתונים בפעם הראשונה!

## הצגת טבלה שלמה

כדי להציג טבלה שלמה, נדרש להזיז את Index שורה אחת קדימה ובכל תזוזה להציג את נתוני השורה בה הוא נמצא. הדרך הבטוחה ביותר לבצע זאת היא באמצעות לולאת do until.

כאשר מתמלא אובייקט Recordset בפעם הראשונה, נמצא Index בתחילת Recordset, כלומר בשורה הראשונה. מקום זה נקרא BOF, כלומר, Beginning Of File. ניתן להתייחס אליו כאל r.bof.

השורה האחרונה מוגדרת כ- End Of File, וניתן להתייחס אליה כאל r.eof

כעת ניצור לולאה אשר תזיז את Index שורה אחת קדימה בכל ריצה, עד אשר יגיע ל- r.eof

תחביר פתיחת הלולאה יהיה:

```
do until r.eof
```

בכל ריצה של הלולאה נציג את נתוני השורה:

```
response.write r.fields("fname") & "---" & r.fields("lname") & "---" & r.fields("email") &
"<br>"
```

ונזיז את Index שורה אחת קדימה על ידי שימוש בשיטה movenext של אובייקט Recordset.

```
r.movenext
```

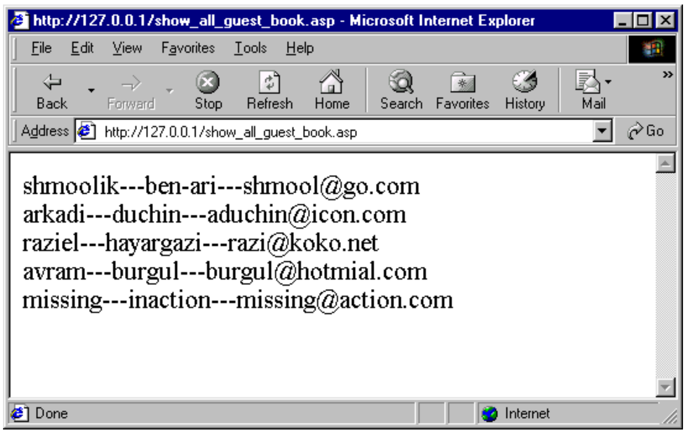
לאחר מכן נסגור את הלולאה על ידי התחביר: loop

כך ייראה הקוד המלא להצגת נתוני כל הטבלה:

show_all_guest_book.asp

```
<%
set c=server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
set r=server.createobject("adodb.recordset")
r.activeconnection=c
r.open "select * from guest_book"
response.write "<font size=5>"
do until r.eof
  response.write r.fields("fname") & "---" & r.fields("lname") & "---" & r.fields("email") &
  "<br>"
  r.movenext
loop
%>
```

וזו תהיה התוצאה :




## הצגת הנתונים בטבלת HTML

כדי להציג את הנתונים בטבלת HTML מסודרת, נדרש לשרשר פקודות טבלת HTML אל תוך הקוד בצורה הבאה :

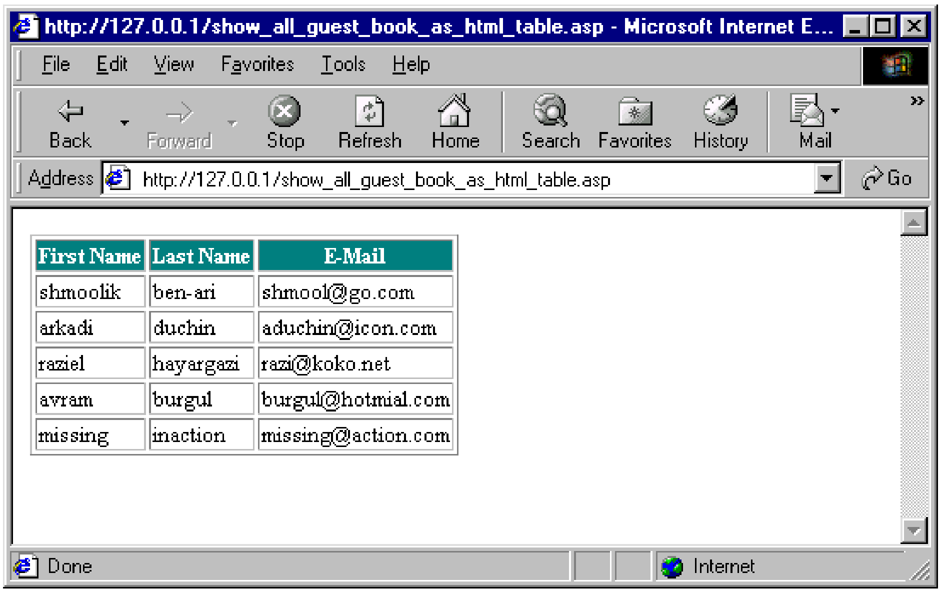
```
show_all_guest_book_as_html_table.asp

<%
set c=server.createobject("adodb.connection")
c.open "dsn=MyDSN;"
set r=server.createobject("adodb.recordset")
r.activeconnection=c
r.open "select * from guest_book"
response.write "<font size=5>"
response.write "<table border>" 'this opens a HTML table
response.write "<tr>" 'this opens a HTML table row
response.write "<th bgcolor=teal><font color=white>First Name</td>" ' First name
'table header
response.write "<th bgcolor=teal><font color=white>Last Name</td>" ' Last name
'table header
response.write "<th bgcolor=teal><font color=white>E-Mail</td>" ' E-Mail table
'header
response.write "</tr>" 'close first table row
do until r.eof
    response.write "<tr>" 'this will open a new HTML table row every run of the loop
    response.write "<td>" & r.fields("fname") & "</td>"
    response.write "<td>" & r.fields("lname") & "</td>"
    response.write "<td>" & r.fields("email") & "</td>"
```

```
    response.write "</tr>" 'this will close the HTML table row every run of the loop
    r.movenext
loop
response.write "</table>" 'this closes the HTML table
%>
```

יש להקפיד כמובן על פתיחת טבלת HTML לפני תחילת הלולאה, וסגירתה לאחר ביצוע הלולאה.

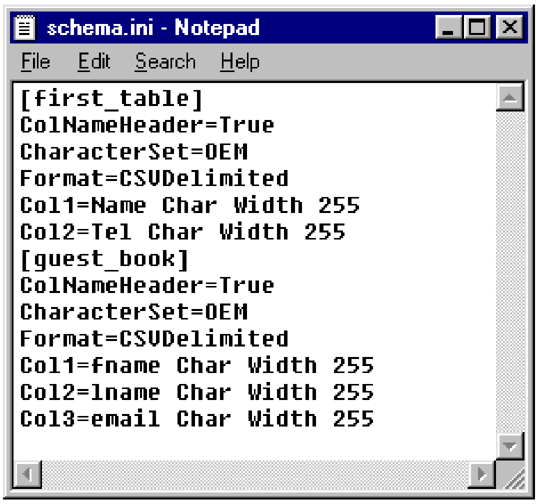
וזו תהיה התוצאה:




בפרק הבא נבנה יישום מבוסס מסד נתונים.

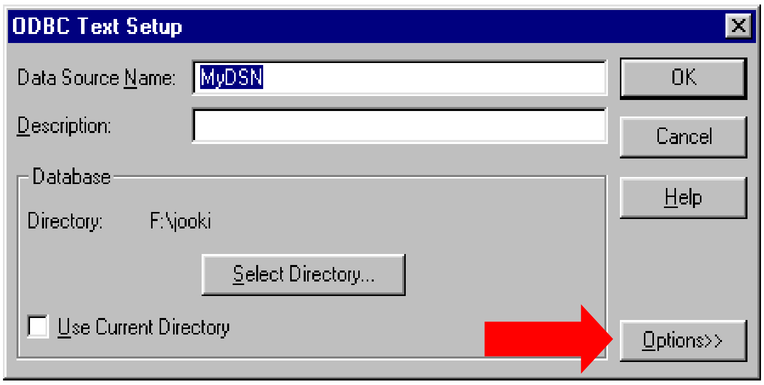
# קובץ Schema.ini

אם תפתחו את הקובץ schema.ini אשר בתיקיית jooki, תגלו כי כאן נמצאות הגדרות הטבלאות. כל שם טבלה מוקף בסוגריים מרובעים ([ ]).

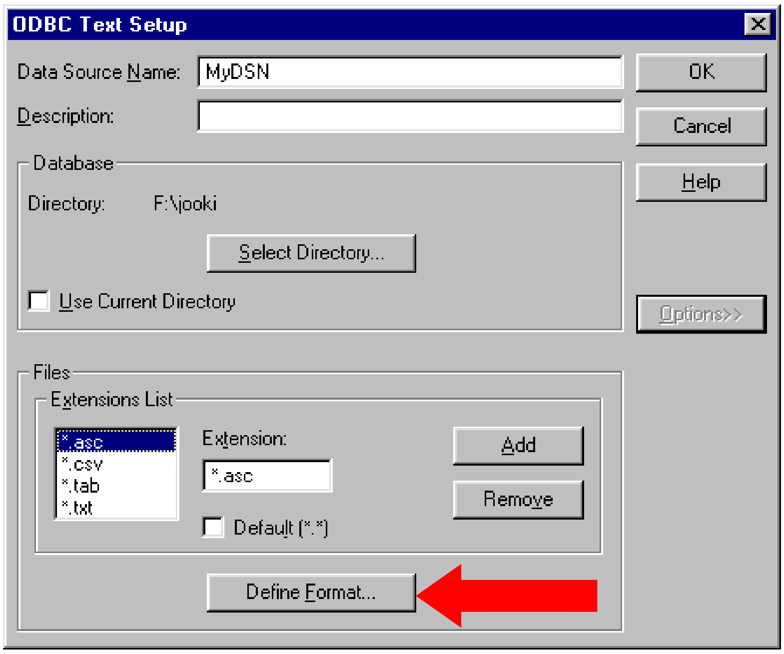


```
[first_table]
ColNameHeader=True
CharacterSet=OEM
Format=CSVDelimited
Col1=Name Char Width 255
Col2=Tel Char Width 255
[guest_book]
ColNameHeader=True
CharacterSet=OEM
Format=CSVDelimited
Col1=fname Char Width 255
Col2=lname Char Width 255
Col3=email Char Width 255
```

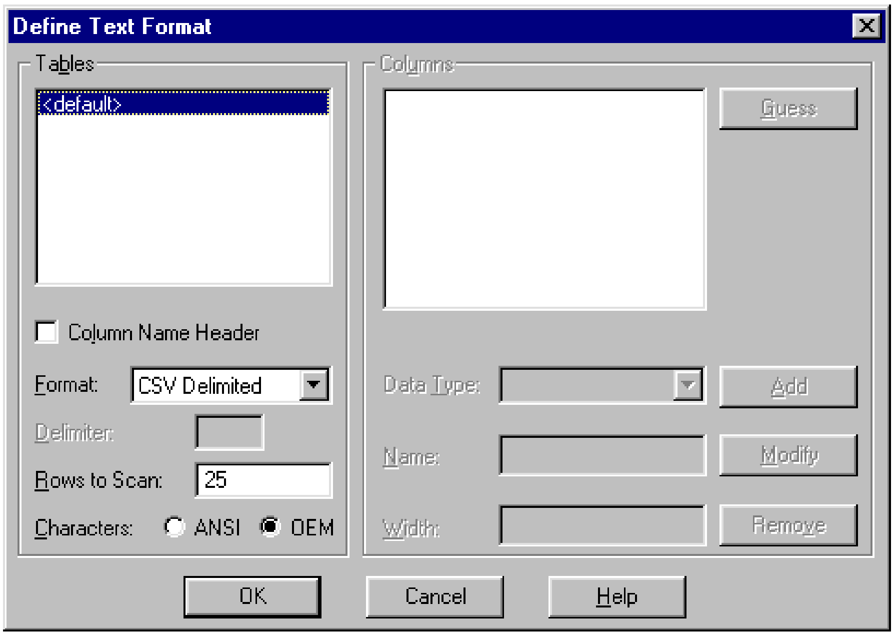
ניתן לראות כי תחת הגדרת first_table, ישנם שני שדות, Name ו- Tel. שניהם מסוג Char ומכילים עד 255 תווים. ניתן לשנות הגדרות אלו באופן ידני דרך פנקס הרשימות, או דרך Microsoft Text Driver ב- ODBC, זאת על ידי לחיצה על לחצן Options במסך הגדרת DSN.



לחיצה על לחצן Options תגדיל את המסך ותאפשר ללחוץ על לחצן Define Format.




לחיצה על לחצן Define Format, תפתח את מסך ההגדרות עבור הטבלאות.




מומלץ בשלב זה לא לשנות הגדרות.

## פרק 11

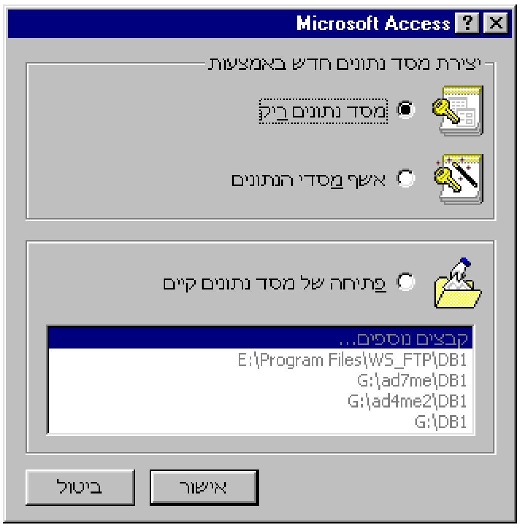
# יישומים מבוססים מסדי נתונים

בפרק זה נבנה יישום מבוסס מסד נתונים, ונבין את הכללים החשובים בבניית יישומים מסוג זה.

הפעם נתייחס למסד הנתונים של מיקרוסופט – **MS Access**. אם אין ברשותך את תוכנת Access, תוכל לבצע את התרגילים על מסדי נתונים מבוססי קבצי טקסט כמו בפרק הקודם. כל שתצטרך הוא להגדיר DSN המשתמש ב- Microsoft Text Driver, וליצור את אותן הטבלאות אשר ניצור ב- Access.

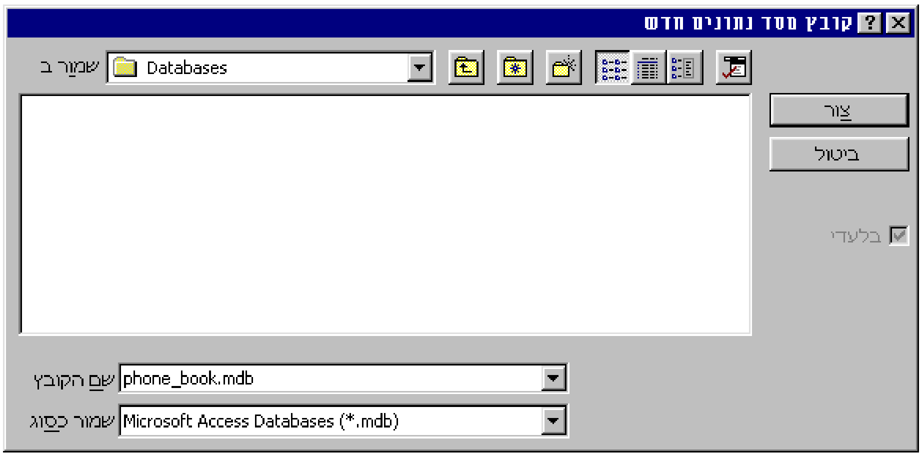
היישום אותו נבנה הוא ספר טלפונים מאובטח. בספר זה יוכלו כולם להביט, אך רק משתמשים מורשים יוכלו לשנות או להוסיף פרטים. לצורך כך נבנה מסד נתונים ב- Access (או בקבצי טקסט כפי שנלמד בפרק הקודם).

נפתח את תוכנת Access ונקבל את המסך הבא:

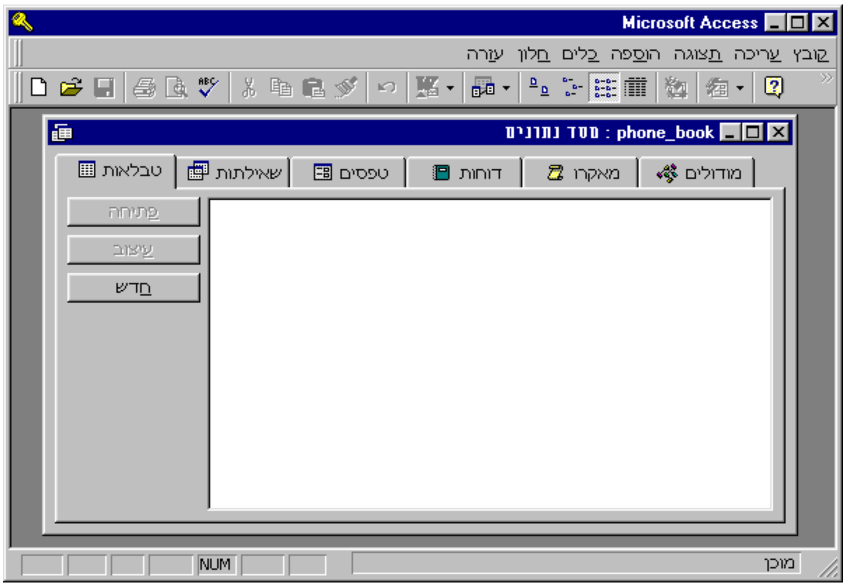



נבחר באפשרות **מסד נתונים ריק** (Empty Database).

המסך הבא שנקבל יהיה זה:

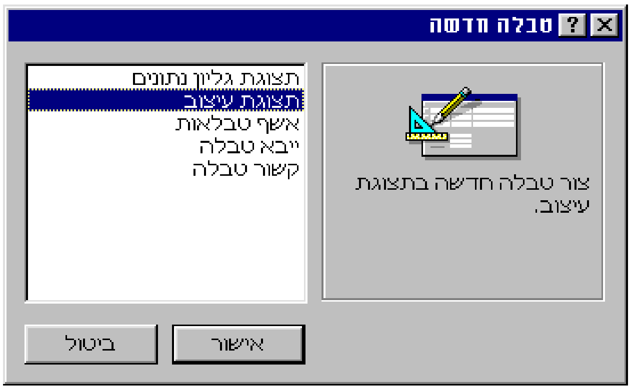



במסך זה נבחר תיקיה לאחסן בה את מסד הנתונים (לצורך התרגיל, פתחו תיקיה חדשה בשם Databases ובחרו בה). העניקו לקובץ את השם phone_book ולחצו על **צור** (Create). זה המסך שיופיע:

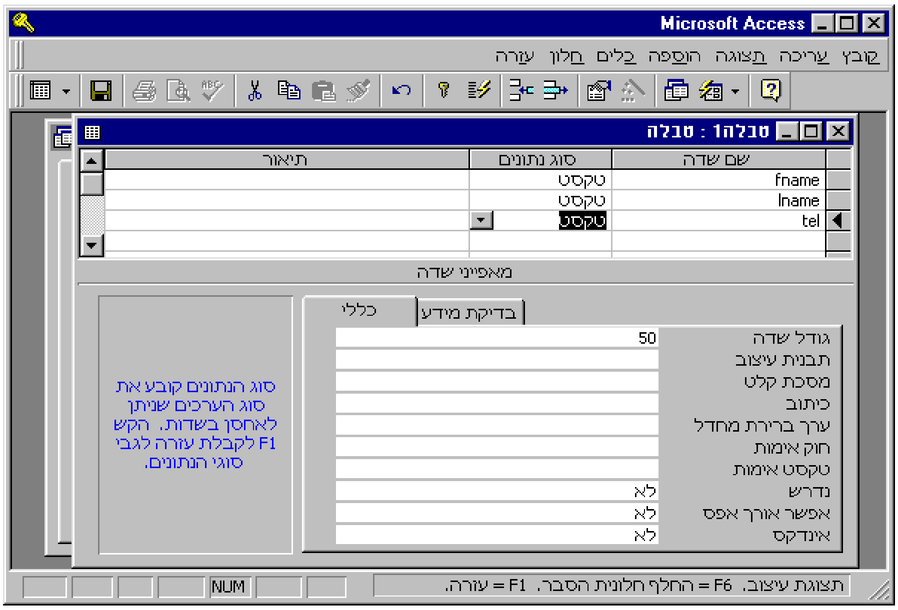



מסך זה מייצג את מסד הנתונים החדש שלנו. במסך זה ניתן לייצר טפסים, דוחות, שאילתות ועוד. כיון שאנו מעוניינים רק ביצירת טבלה, נישאר בלשונית הטבלאות, ונלחץ על **חדש** (New) כדי ליצור טבלה חדשה.

נקבל את המסך הבא:




במסך זה נבחר **תצוגת עיצוב** (Design View). נקבל את המסך הבא:




מסך זה מאפשר הגדרה של עמודות בטבלה החדשה שניצור, הכוללת: שם עמודה, סוג עמודה (טקסט, מספר וכו') והערות לגבי העמודה.

במסך זה נגדיר **חמש** עמודות:

fname – מסוג טקסט

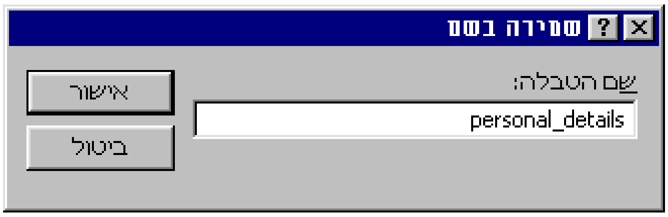
lname – מסוג טקסט

tel – מסוג טקסט

address – מסוג טקסט

occupation – מסוג טקסט

לאחר מכן נבחר בתפריט **קובץ←שמור** (save<file) ונעניק את השם personal_details לטבלה באופן הבא:

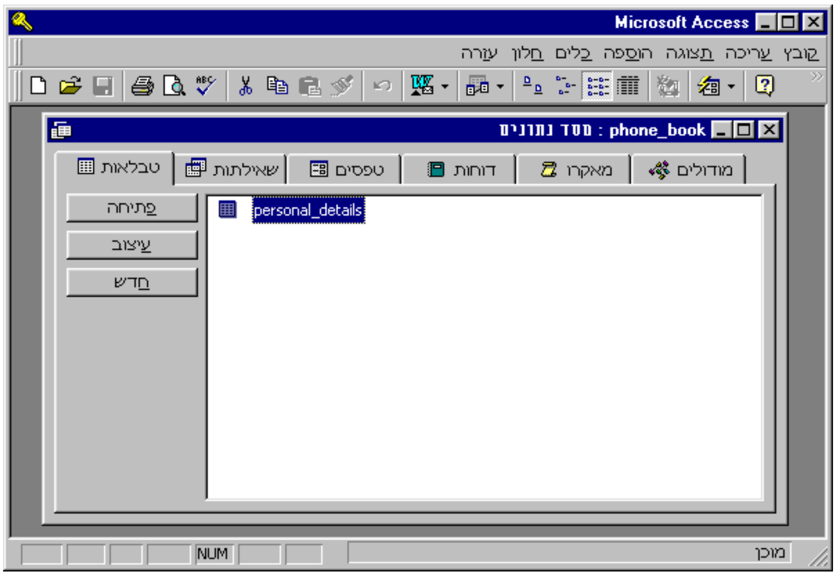



Access יודיע את ההודעה הבאה:




תיבה זו מציעה לצרף עמודה לטבלה אשר תכיל מספר חד ערכי לכל שורה. ערך חד ערכי כזה נקרא בטבלה **מפתח ראשי** (primary key), ומשמש לזיהוי חד ערכי ברשומות. כיון שאנו לא נשתמש במפתח חד ערכי ביישום הבא, נלחץ על **לא** (No).

כעת, נסגור את החלון המכיל את הגדרות הטבלה (היזהרו לא לסגור את חלון Access הגדול!), ונקבל את המסך הבא, המראה כי במסד הנתונים שיצרנו (phone_book), יש טבלה אחת בשם Personal_details.

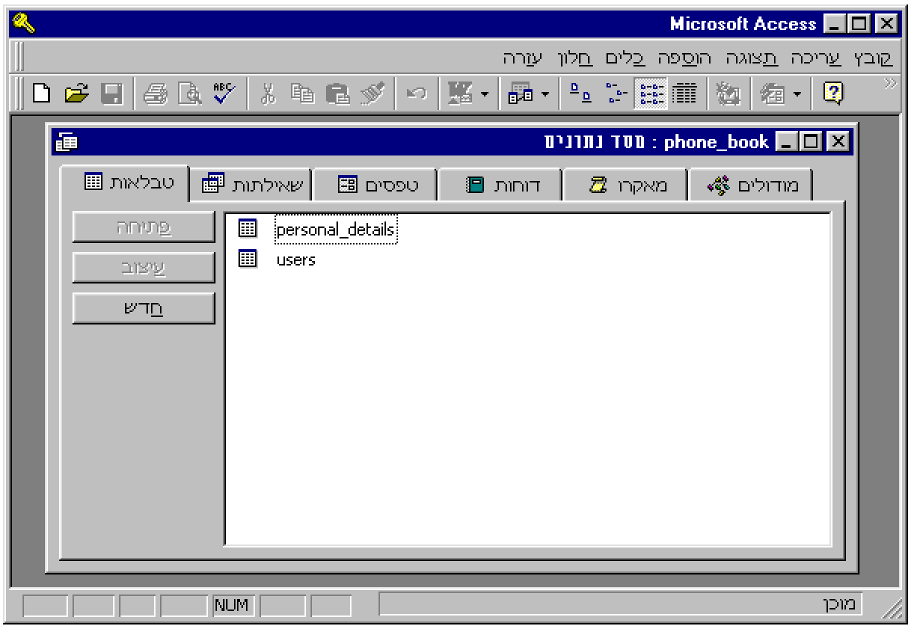



כעת, נחזור על תהליך יצירת הטבלה ונגדיר טבלה חדשה בעלת שתי עמודות:

user – מסוג טקסט

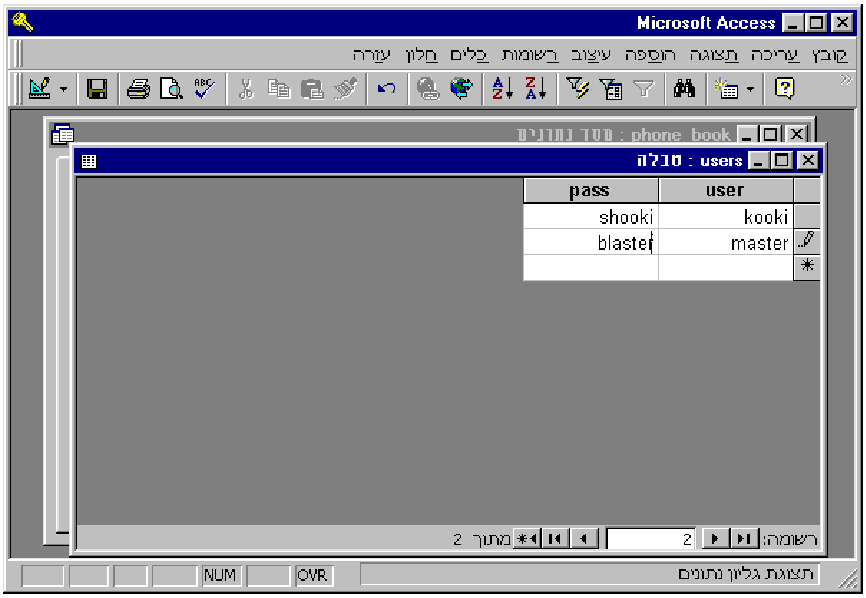
pass – מסוג טקסט

ונקרא לה users. אם ביצעתם את התהליך כהלכה, תקבלו את המסך הבא:



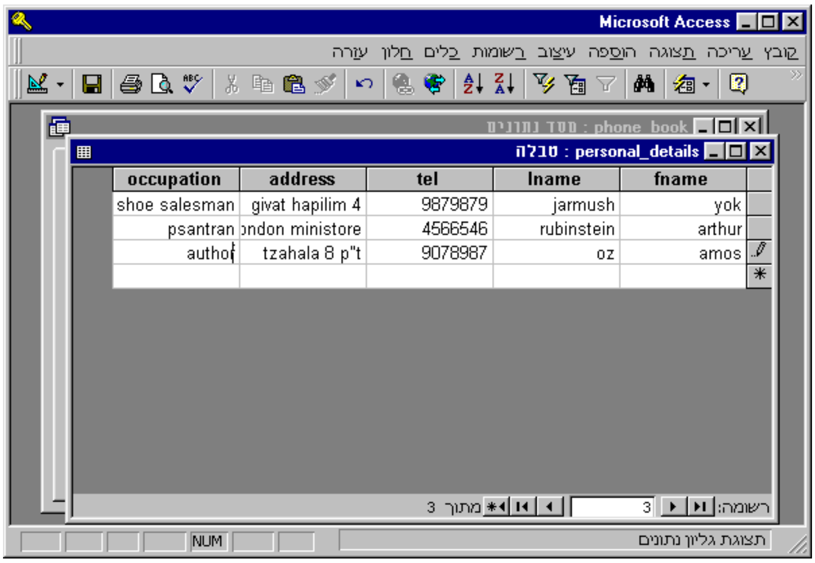

כעת, לחצו על טבלת users בעזרת הלחצן הימני של העכבר, ובחרו באפשרות **פתיחה** (Open).

המסך שיתקבל:



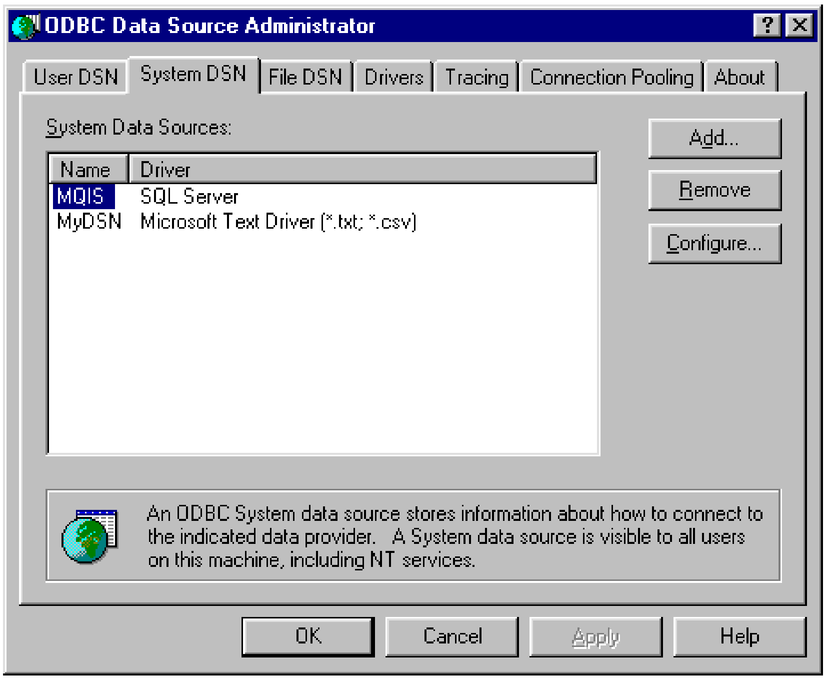

מסך זה מאפשר הזנת נתונים אל הטבלה הנבחרת. הזינו שני שמות משתמש ושתי סיסמאות. בחרו בתפריט **קובץ←שמור** (save<file) וסגרו את החלון.

נבצע את אותו תהליך לגבי טבלת personal_details (לחצן ימני על שם הטבלה ובחירה באפשרות **פתיחה**) ונזין לפחות 3 רשומות (**שונות!**) באופן הבא:

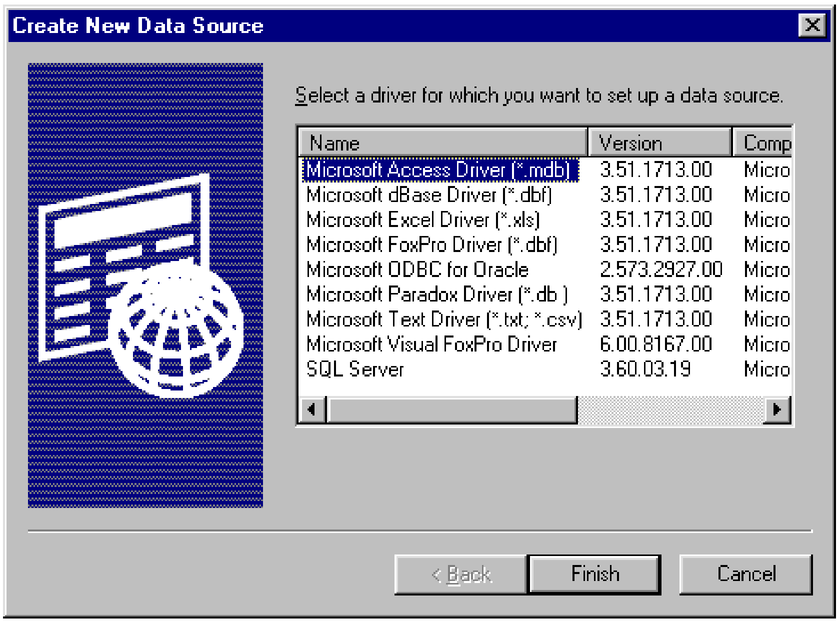


**שימו לב:** למעשה, יצרנו שתי טבלאות. טבלה ראשונה, טבלת personal_details, תשמש אותנו לשמירת נתוני ספר הטלפונים. טבלה שנייה, טבלת users, תשמש אותנו לניהול משתמשים מורשים לעדכון ספר הטלפונים.

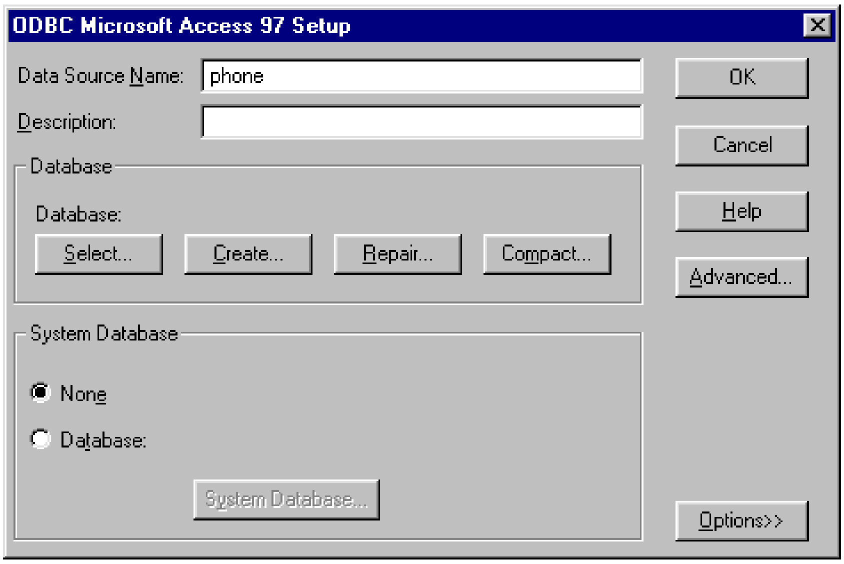
כל שנותר לבצע כעת, הוא להגדיר DSN ב-ODBC, כדי שנוכל לגשת למסד הנתונים דרך אובייקטים של ADO מ-ASP. נפתח את ODBC (דרך לוח הבקרה – control pannel) ונבחר בלשונית **System DSN** כפי שביצענו בפרק הקודם:



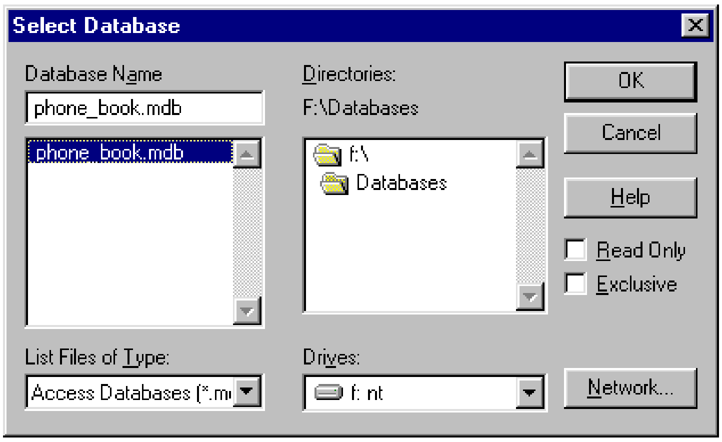
נלחץ על לחצן **הוסף** (Add) ונבחר **Microsoft Access Driver**.




לאחר שנלחץ על **סיים** (Finish), נקבל את המסך הבא :



בשדה **Data Source Name** נכתוב את שם DSN כ- phone ואז נלחץ על לחצן **בחר** (Select).




בחלון שנפתח, נבחר את קובץ מסד הנתונים שיצרנו ב- Access ← phone_book.mdb, נלחץ **אישור** (OK). עתה נלחץ שוב **אישור** (OK) בשני המסכים הבאים כדי לאשר את פעולת ההגדרה שביצענו. כעת, נוכל לבקש לגשת ל- DSN בשם phone, ולקבל גישה אל מסד הנתונים phone_book.mdb, המכיל את שתי הטבלאות איתן נעבוד, personal_details ו- users.

# תחילת בנייה של היישום

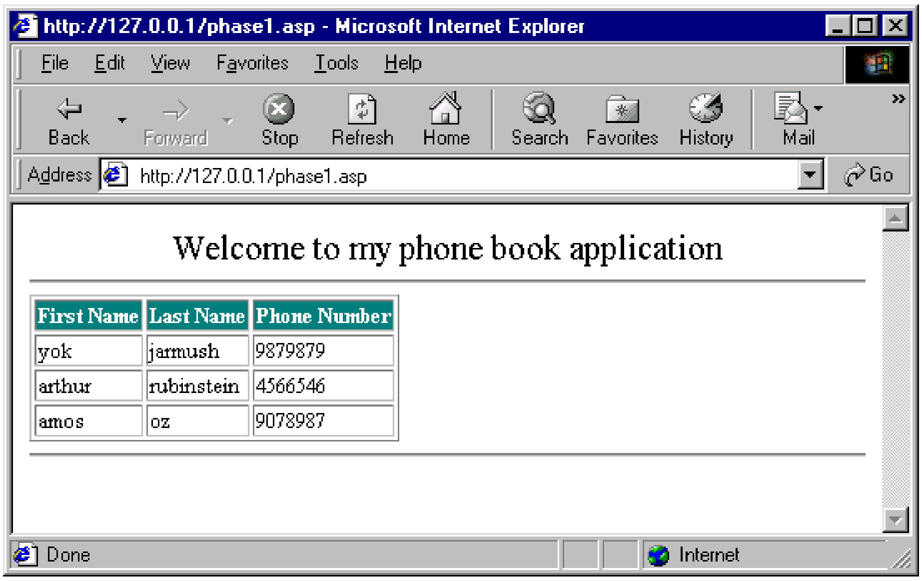
נבנה את מסמך ASP השולף נתונים ומציג אותם כטבלה. כמובן, נשתמש באובייקטים Connection ו- Recordset, כפי שלמדנו בפרק הקודם.

Phase1.asp

```
<%
Set c=server.createobject("adodb.connection")
c.open "dsn=phone;"
set r=server.createobject("adodb.recordset")
r.activeconnection=c
r.open "select * from personal_details"
response.write "<body>"
response.write "<font size=5><center>Welcome to my phone book
application</center></font>"
response.write "<hr>"
response.write "<table border><tr>"
response.write "<th bgcolor=teal><font color=white>First Name</th>"
response.write "<th bgcolor=teal><font color=white>Last Name</th>"
response.write "<th bgcolor=teal><font color=white>Phone Number</th>"
response.write "</tr>"
```

```
do until r.eof
        response.write "<tr>"
        response.write "<td>" & r.fields("fname") & "</td>"
        response.write "<td>" & r.fields("lname") & "</td>"
        response.write "<td>" & r.fields("tel") & "</td>"
        response.write "</tr>"
        r.movenext
loop
repsonse.write "</table><hr>"
%>
```

כך ייראה השלב הראשון בבניית היישום:



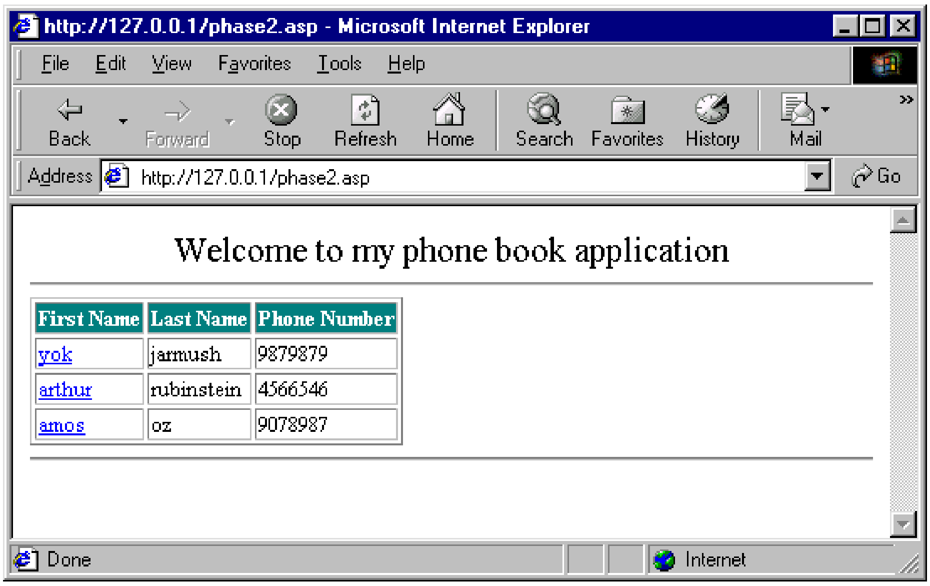

שימו לב, כי לא מוצגים כל הנתונים השמורים בטבלה. זאת כדי לאפשר הצגת השיטה של הפיתוח המאפשרת מעבר בין דפי יישום תוך שימוש בקישורים. כלומר, בסיומו, יאפשר מסך זה לצפות בנתונים הבסיסיים (שם פרטי, שם משפחה וטלפון), ועל ידי לחיצה על אחד השמות הפרטיים, נקבל מסך פרטים אישי עבור אותו אדם, ובו נוכל לראות את כל הפרטים השמורים בטבלה, לרבות כתובת ועיסוק.

בגלגול הבא של המסמך, נוסיף לכל הצגה של שם פרטי את התג <a href=personal_page.asp> כדי לאפשר למשתמש להגיע לפרטים האישיים של כל מי שרשום בטבלה באופן הבא:

phase2.asp

```
<%
Set c=server.createobject("adodb.connection")
c.open "dsn=phone;"
set r=server.createobject("adodb.recordset")
r.activeconnection=c
r.open "select * from personal_details"
response.write "<body>"
response.write "<font size=5><center>Welcome to my phone book
application</center></font>"
response.write "<hr>"
response.write "<table border><tr>"
response.write "<th bgcolor=teal><font color=white>First Name</th>"
response.write "<th bgcolor=teal><font color=white>Last Name</th>"
response.write "<th bgcolor=teal><font color=white>Phone Number</th>"
response.write "</tr>"
do until r.eof
        response.write "<tr>"
        response.write "<td><a href=personal_page.asp>" &_
        r.fields("fname") & "</a></td>"
        response.write "<td>" & r.fields("lname") & "</td>"
        response.write "<td>" & r.fields("tel") & "</td>"
        response.write "</tr>"
        r.movenext
loop
response.write "</table><hr>"
%>
```

התוצאה תיראה כך :



# בעיה!

כרגע, כל שם פרטי הינו קישור. עם לחיצה על כל שם פרטי, נגיע לקובץ personal_page.asp. אולם, כיוון שכל הקישורים זהים (היינו, <a href=personal_page.asp>), לא נוכל להחליט את הפרטים של איזה אדם להציג.

הפתרון מגיע בעזרת השיטה get המאפשרת שליחת פרמטרים בשורת הכתובת. כך שבמקום לרשום:

```
response.write "<td><a href=personal_page.asp>" &_
r.fields("fname") & "</a></td>"
```

נרשום:

```
response.write "<td><a href=personal_page.asp?p=" &_
r.fields("fname") & ">" &_
r.fields("fname") & "</a></td>"
```

כיון ששליפת המידע והצגתו מתבצעת בלולאה, תהפוך השורה מצורה זו לצורה הבאה:

<td><a href=personal_page.asp?p=**yok>yok**</a></td>
<td><a href=personal_page.asp?p=**arthue>arthur**</a></td>
<td><a href=personal_page.asp?p=**amos>amos**</a></td>

כך שבקובץ personal_page.asp, נוכל לגזור את הפרמטר p ולדעת לאיזה אדם התכוון המשתמש כאשר לחץ על הקישור.

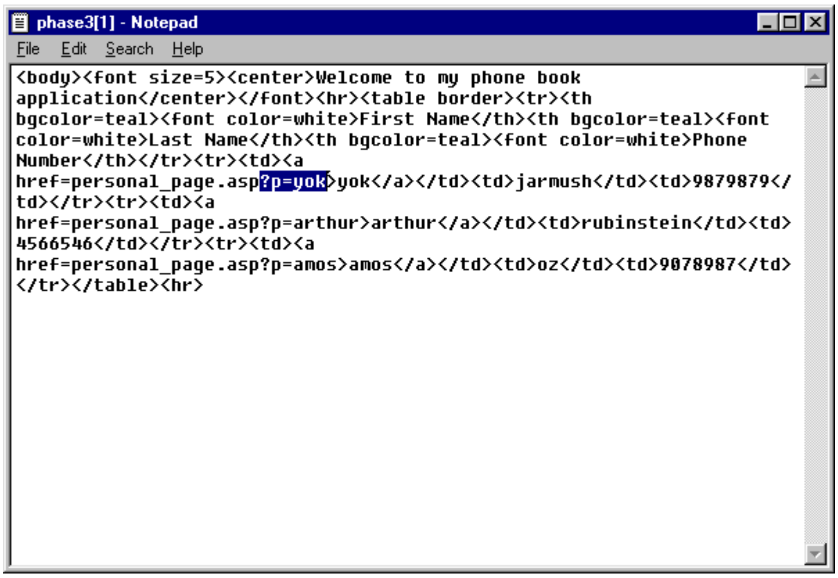
קוד זה:

phase3.asp

```
<%
Set c=server.createobject("adodb.connection")
c.open "dsn=phone;"
set r=server.createobject("adodb.recordset")
r.activeconnection=c
r.open "select * from personal_details"
response.write "<body>"
response.write "<font size=5><center>Welcome to my phone book
application</center></font>"
response.write "<hr>"
response.write "<table border><tr>"
response.write "<th bgcolor=teal><font color=white>First Name</th>"
response.write "<th bgcolor=teal><font color=white>Last Name</th>"
response.write "<th bgcolor=teal><font color=white>Phone Number</th>"
response.write "</tr>"
```

```
do until r.eof
        response.write "<tr>"
        response.write "<td><a href=personal_page.asp?p=" &_
        r.fields("fname") &_
        ">" & r.fields("fname") & "</a></td>"
        response.write "<td>" & r.fields("lname") & "</td>"
        response.write "<td>" & r.fields("tel") & "</td>"
        response.write "</tr>"
        r.movenext
loop
response.write "</table><hr>"
%>
```

ייתן לכאורה את אותה תוצאה כמו phase2, אך כאשר נבקש view source בדפדפן, נראה כי כל קישור שולח איתו נתונים שונים. במקרה זה את שמו הפרטי של האדם.

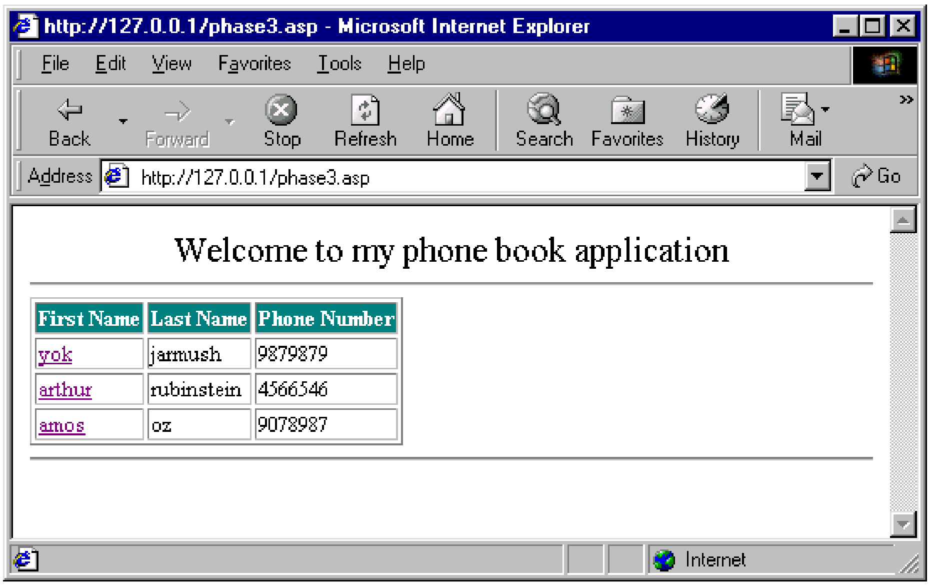


כעת, נכין את קובץ personal_page.asp. בקובץ זה, נבקש את ערך הפרמטר **p** המכיל את שם האדם עליו ביקש המשתמש לקבל את המידע, ונחלץ את הנתונים הרשומים על אותו אדם במסד הנתונים. למעט העובדה כי לא נשתמש בלולאה משום שאין צורך ביותר משורה אחת בטבלה, ולסגירת אובייקט Recordset, אין קוד זה מכיל כל חידוש.

Personal_page.asp

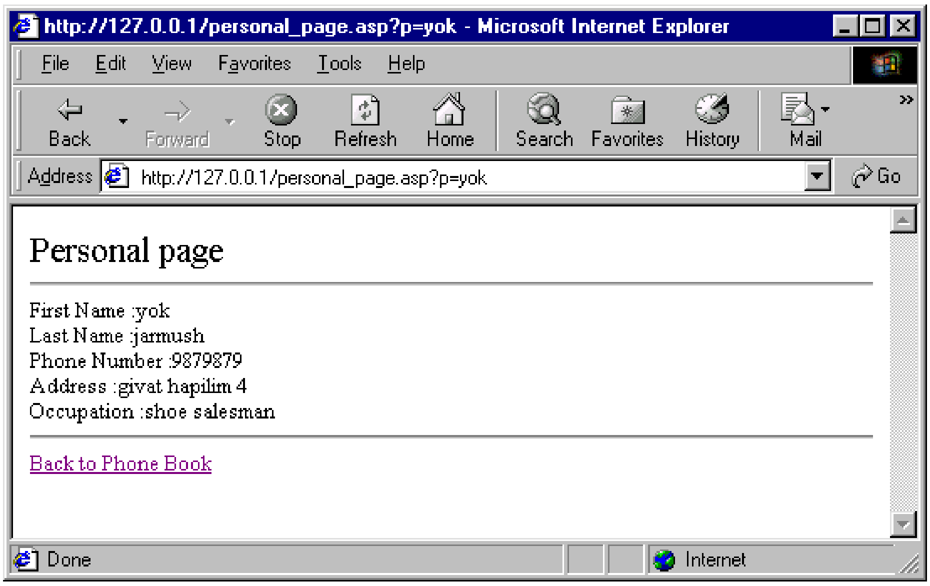
```
<%
set c=server.createobject("adodb.connection")
c.open "dsn=phone;"
set r=server.createobject("adodb.recordset")
r.activeconnection=c
r.open "select * from personal_details where fname='" &_
 request.querystring("p") & "'"
response.write "<font size=5>Personal page</font><hr>"
response.write "First Name :" & r.fields("fname") & "<br>"
response.write "Last Name :" & r.fields("lname") & "<br>"
response.write "Phone Number :" & r.fields("tel") & "<br>"
response.write "Address :" & r.fields("address") & "<br>"
response.write "Occupation :" & r.fields("occupation") & "<hr>"
response.write "<a href=phase3.asp>Back to Phone Book</a>"
%>
```

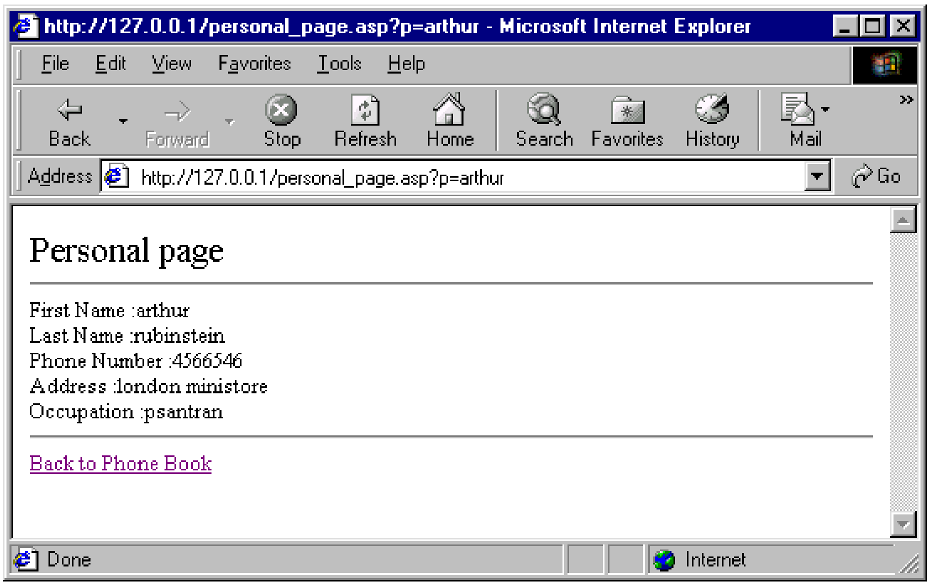
כעת, נפעיל את 127.0.0.1/phase3.asp ונקבל את המסך הבא:

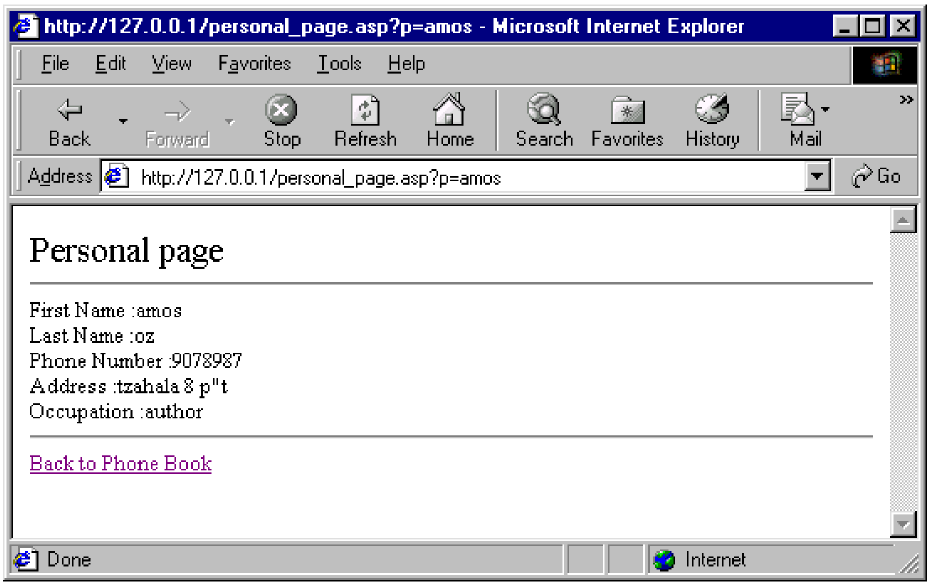


כאשר נלחץ על כל אחד מהשמות הפרטיים (First Name), נקבל מסך אישי המתאים לו בשל העובדה שכל קישור שולח את הפרמטר **p** עם ערך אחר. כאן אולי המקום להסביר, כי בשל העובדה שאין שני שמות פרטיים זהים, ניתן לזהות אדם באופן חד ערכי על פי שמו. ביישום מציאותי, יש צורך ב**מפתח ראשי** (עמודה המכילה מספר חד ערכי עבור כל רשומה ומאפשרת זיהוי חד ערכי).

להלן 3 המסכים האישיים של שלוש הרשומות בטבלה. שימו לב לשורת הכתובת בכל אחד מהם :






כאן, בעזרת השימוש בקוד פתוח, יצרנו 3 דפי HTML שונים באמצעות קובץ ASP אחד. כמובן, קוד זה יכול לייצר אלפי דפי HTML שונים, בהתאם לאנשים הרשומים בטבלה.

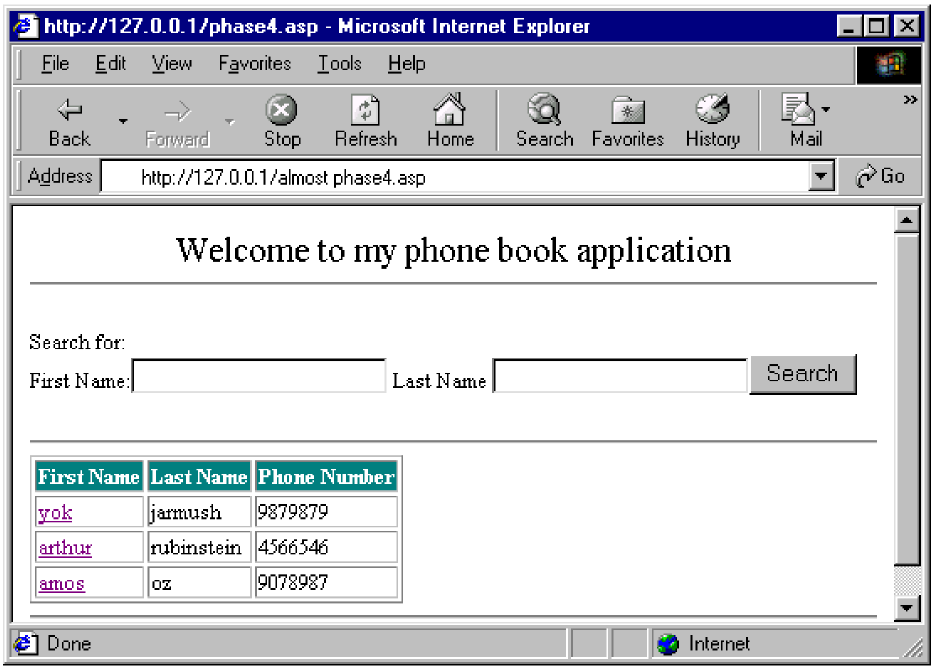
# חיפוש

כעת נוסיף למסך הצגת הטבלה, אפשרות חיפוש מספר טלפון על פי שם. נאפשר למשתמש להקיש שם פרטי, או שם משפחה, או את שניהם. כדי לאפשר חיפוש, נוסיף לקוד הקיים, טופס HTML המכיל שני שדות, עבור שם פרטי ושם משפחה באופן הבא (כמו כן, נעביר את תגית body ואת הכותרת ל- HTML רגיל במקום response.write):

```
<body>
<font size=5><center>Welcome to my phone book application</center></font>
<hr>
<form>
Search for:
First Name:<input type=text name=fn> Last Name <input type=text name=ln><input
type=submit value=Search></form>
```

מכיון שלא נשתמש בספר זה בבדיקות בצד לקוח באמצעות JavaScript, לא ניתן שם לטופס.

הוספת קוד זה כמות שהוא למסמך phase3.asp תיתן את התוצאה הבאה:




ניתן לשים לב, כי לטופס אין תכונת Action. עובדה זו, תשלח את נתוני הטופס לאותו ASP ממנו נוצר. זו הזדמנות טובה לנצל את תפיסת Page Reentry המופיעה בפרק 5 על אובייקט Response.

בתפיסה זו, כל עוד לא נשלחו אל ASP פרמטרים (סימן לכך שהקובץ עולה בפעם הראשונה), יוצגו נתונים מסוימים. אם נשלחו פרמטרים (סימן לכך שהקובץ עולה בשנית לאחר שנלחץ ונשלח מתוכו טופס), יוצגו נתונים אחרים. אם לא נשלחו פרמטרים (המסמך עולה בפעם הראשונה), נציג את הטופס והטבלה כפי שהיא מוצגת בצילום המסך הנ"ל. אם נשלחו פרמטרים (המסך עולה שוב לאחר שנלחץ לחצן החיפוש), נציג את אותו מסך בתוספת תשובת החיפוש. אם כן, נבדוק אם נשלחו פרמטרים. למעשה נבדוק אם הפרמטרים אינם ריקים על ידי התחביר:

```
if not isempty(request.querystring("fname")) then
```

בדיקה זו תוודא כי הפרמטרים אינם ריקים (not isempty) וכך נדע כי הגולש הקיש על לחצן Submit ומעוניין לקבל תוצאת חיפוש. כדי לייעל את הקוד, נכריז על אובייקט Connection ו- Recordset בתחילת המסמך, וכך נוכל להשתמש בהם במהלך הבדיקה.

כך יתבטא רעיון זה בקוד:

```
if not isempty(request.querystring("fname")) then
        r.open "select tel from personal_details where fname='" &_
       request.queystring("fn") &_
        "' and lname='" & request.querystring("ln") & "'"
        response.write "<br>Phone Number : " & r.fields("tel")
end if
```

בקוד זה שתי בעיות משמעותיות. האחת, אין טיפול נפרד במקרה והמשתמש הקיש רק שם פרטי או רק שם משפחה. שתיים, אין טיפול במקרה והמשתמש הקיש שם שאינו קיים בטבלה.

# טיפול בשאילתה שגויה – שם שאינו קיים

תחילה, נטפל במקרה בו המשתמש הקיש שם שאינו קיים בטבלה. במצב בו אובייקט Recordset ביצע שאילתה שאינה מחזירה כל תשובה (במקרה של חיפוש שם שאינו קיים), נוצר מצב בו Index נמצא כהרגלו בתחילת BOF) Recordset), אך מכיון שאין כל רשומות, הוא נמצא גם בסוף EOF) Recordset). כל שעלינו לכתוב כדי לברר אם הרשומה ריקה, הוא:

```
if r.eof then
```

כלומר לבצע בדיקה לוגית המוודאת אם Index נמצא בסוף Recordset מייד לאחר השאילתה, מצב המתקיים רק אם השאילתה לא החזירה תשובה.

בדיקת הגשת שם פרטי ו/או שם משפחה:

תהליך הבדיקה שנבצע הוא פשוט. נשאל תחילה אם השם הפרטי שווה ל- "", כלומר, ריק. אם השם הפרטי ריק, נבדוק אם שם המשפחה אינו ריק גם הוא, ונבצע שאילתה על פי שם המשפחה בלבד. אם שם המשפחה קיים, נבדוק אם גם השם הפרטי קיים, ונבצע שאילתה בהתאם. כך ייראה הקוד (הכולל גם את הטיפול בשאילתה שגויה):

```
if not isempty(request.querystring("fn")) then ' this if only checks if the file runs for the
first time.
        If request.querystring("fn")="" then
          If request.querystring("ln")="" then
            Response.write "<b>You did not type a person name!"
           Else
            r.open "select * from personal_details where lname='" &_
           request.querystring("ln") & "'"
             If r.eof then
               Response.write "<b>The person you asked for is not listed!"
               r.close
             Else
               Response.write "First Name : " & r.fields("fname") & "<br>"
```

```
            Response.write "Last Name : " & r.fields("lname") & "<br>"
            Response.write "Phone Number : " & r.fields("tel")
            r.close
          End if
        End if
      Else
        If request.querystring("ln")="" then
            r.open "select * from personal_details where fname='" &_
            request.querystring("fn") & "'"
            if r.eof then
              Response.write "<b>The person you asked for is not listed!"
              r.close
            Else
             Response.write "First Name : " & r.fields("fname") & "<br>"
             Response.write "Last Name : " & r.fields("lname") & "<br>"
             Response.write "Phone Number : " & r.fields("tel")
              r.close
            End if
        Else
            r.open "select * from personal_details where fname='" &_
            request.querystring("fn") &_
            "' and lname='" & request.querystring("ln") & "'"
            if r.eof then
             Response.write "<b>The person you asked for is not listed!"
             r.close
            Else
             Response.write "First Name : " & r.fields("fname") & "<br>"
             Response.write "Last Name : " & r.fields("lname") & "<br>"
             Response.write "Phone Number : " & r.fields("tel")
             r.close
            End if
          End if
        End if
      End if
```

אלגוריתם מסורבל זה עובד, ותוכלו למוצאו בקובץ הבא :

phase4.asp

```
<%
Set c=server.createobject("adodb.connection")
c.open "dsn=phone;"
set r=server.createobject("adodb.recordset")
r.activeconnection=c
%>
```

```
<body>
<font size=5><center>Welcome to my phone book application</center></font>
<hr>
<form>
Search for:<br>
First Name:<input type=text name=fn> Last Name <input type=text name=ln><input
type=submit value=Search></form>
<%
if not isempty(request.querystring("fn")) then ' this if only checks if the file runs for the
'first time.
        If request.querystring("fn")="" then
          If request.querystring("ln")="" then
            Response.write "<b>You did not type a person name!"
           Else
            r.open "select * from personal_details where lname='" &_
           request.querystring("ln") & "'"
             If r.eof then
               Response.write "<b>The person you asked for is not listed!"
                     r.close
             Else
               Response.write "First Name : " & r.fields("fname") & "<br>"
               Response.write "Last Name : " & r.fields("lname") & "<br>"
               Response.write "Phone Number : " & r.fields("tel")
               r.close
             End if
           End if
        Else
          If request.querystring("ln")="" then
              r.open "select * from personal_details where fname='" &_
              request.querystring("fn") & "'"
              if r.eof then
                Response.write "<b>The person you asked for is not listed!"
                r.close
              Else
               Response.write "First Name : " & r.fields("fname") & "<br>"
               Response.write "Last Name : " & r.fields("lname") & "<br>"
               Response.write "Phone Number : " & r.fields("tel")
               r.close
              End if
           Else
              r.open "select * from personal_details where fname='" &_
               request.querystring("fn") &_
              "' and lname='" & request.querystring("ln") & "'"
```

```
            if r.eof then
              Response.write "<b>The person you asked for is not listed!"
              r.close
            Else
              Response.write "First Name : " & r.fields("fname") & "<br>"
              Response.write "Last Name : " & r.fields("lname") & "<br>"
              Response.write "Phone Number : " & r.fields("tel")
              r.close
            End if
          End if
        End if
      End if
%>
<hr>
<%
r.open "select * from personal_details"

response.write "<table border><tr>"
response.write "<th bgcolor=teal><font color=white>First Name</th>"
response.write "<th bgcolor=teal><font color=white>Last Name</th>"
response.write "<th bgcolor=teal><font color=white>Phone Number</th>"
response.write "</tr>"
do until r.eof
      response.write "<tr>"
      response.write "<td><a href=personal_page.asp?p=" & r.fields("fname") &_
       ">" & r.fields("fname") & "</a></td>"
      response.write "<td>" & r.fields("lname") & "</td>"
      response.write "<td>" & r.fields("tel") & "</td>"
      response.write "</tr>"
      r.movenext
loop
response.write "</table><hr>"
%>
```

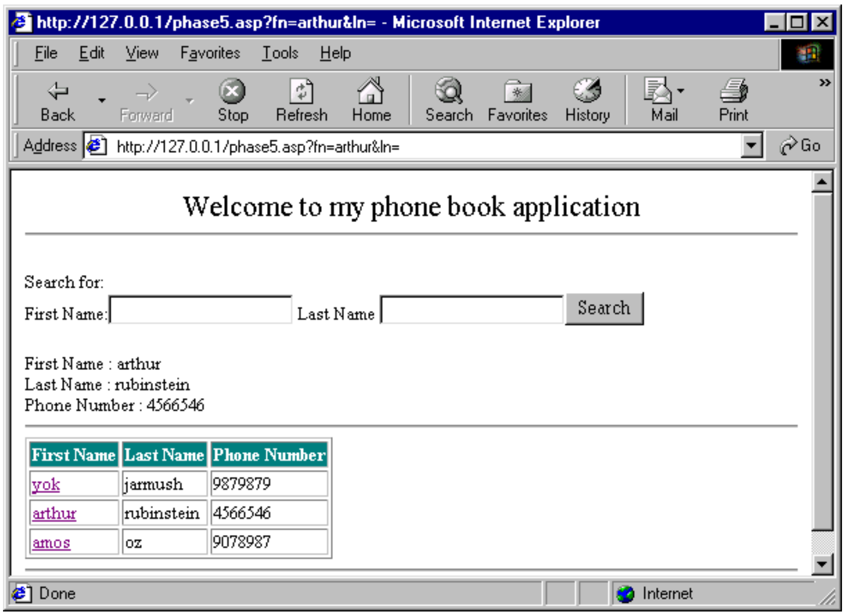
אמנם, קוד זה עובד, אך הוא מסורבל ולא יעיל. שימו לב, כי לאחר כל פתיחת אובייקט Recordset, חייבת להתבצע סגירה כדי לאפשר גישות נוספות למסד הנתונים. כמו כן, שורות קוד רבות חוזרות על עצמן. ניתן לפתור בעיה זו על ידי בדיקת צד לקוח באמצעות JavaScript אשר בנוסף לפרמטרים, ישלח דגל המודיע אילו שדות מולאו. על פי דגל זה תבוצע השאילתה.

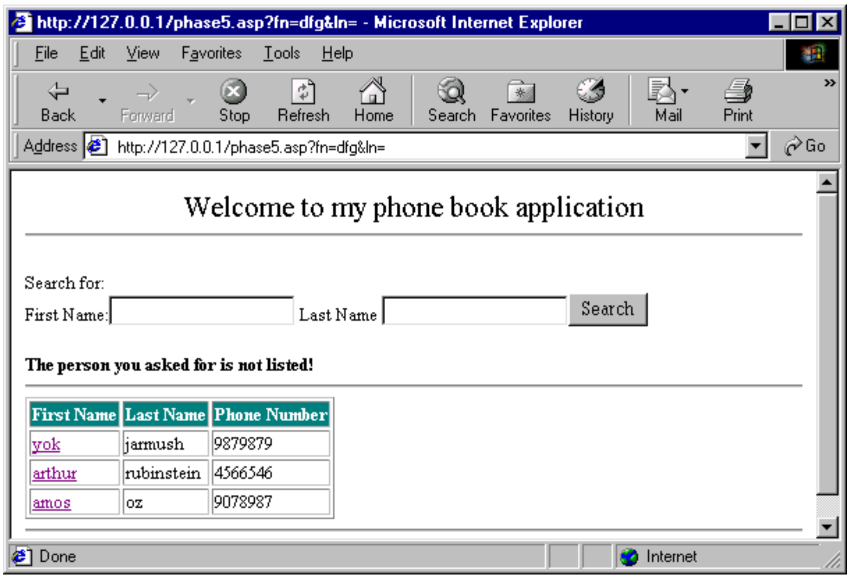
ספר זה דן בתכנות בצד השרת, ולכן נייעל קוד זה בצורה אחרת. נבצע את הבדיקות ונקבע את ערך המשתנה המחרוזתי. לאחר סיום הבדיקות, נבצע את השאילתה המתאימה למסד הנתונים תוך שימוש במחרוזת השאילתה המתאימה.

לדוגמה:

```
if not isempty(request.querystring("fn")) then
 if request.querystring("fn")="" then
   if request.querystring("ln")<>"" then
       type_of_query=" lname='" & request.querystring("ln") & "'"
   end if
 else
   if request.querystring("ln")="" then
       type_of_query=" fname='" & request.querystring("fn") & "'"
   else
       type_of_query=" fname='" & request.querystring("fn") & "' and lname='" &_
       request.querystring("ln") & "'"
   end if
 end if
if isempty(type_of_query) then
    response.write "<b>You did not type a person name!"
 else
    r.open "select * from personal_details where" & type_of_query
    if r.eof then
       response.write "<b> The person you asked for is not listed!"
r.close
    else
       Response.write "First Name : " & r.fields("fname") & "<br>"
       Response.write "Last Name : " & r.fields("lname") & "<br>"
       Response.write "Phone Number : " & r.fields("tel")
       r.close
    end if
 end if
end if
```

כך ייראו תשובות שונות לחיפושים שונים:



:לפניכם הקוד המלא

phase5.asp

```
<%
Set c=server.createobject("adodb.connection")
c.open "dsn=phone;"
set r=server.createobject("adodb.recordset")
r.activeconnection=c
%>
<body>
<font size=5><center>Welcome to my phone book application</center></font>
<hr>
<form>
Search for:<br>
First Name:<input type=text name=fn> Last Name <input type=text name=ln><input
type=submit value=Search></form>
<%
if not isempty(request.querystring("fn")) then
 if request.querystring("fn")="" then
   if request.querystring("ln")<>"" then
       type_of_query=" lname='" & request.querystring("ln") & "'"
   end if
 else
   if request.querystring("ln")="" then
       type_of_query=" fname='" & request.querystring("fn") & "'"
   else
       type_of_query=" fname='" & request.querystring("fn") & "' and lname='" &_
       request.querystring("ln") & "'"
   end if
 end if

 if isempty(type_of_query) then
    response.write "<b>You did not type a person name!"
 else
    r.open "select * from personal_details where" & type_of_query
    if r.eof then
       response.write "<b> The person you asked for is not listed!"

       r.close
    else
       Response.write "First Name : " & r.fields("fname") & "<br>"
       Response.write "Last Name : " & r.fields("lname") & "<br>"
       Response.write "Phone Number : " & r.fields("tel")
```

```
      r.close
    end if
  end if
end if

%>
<hr>
<%
r.open "select * from personal_details"
response.write "<table border><tr>"
response.write "<th bgcolor=teal><font color=white>First Name</th>"
response.write "<th bgcolor=teal><font color=white>Last Name</th>"
response.write "<th bgcolor=teal><font color=white>Phone Number</th>"
response.write "</tr>"
do until r.eof
        response.write "<tr>"
        response.write "<td><a href=personal_page.asp?p=" & r.fields("fname") &_
 ">" & r.fields("fname") & "</a></td>"
        response.write "<td>" & r.fields("lname") & "</td>"
        response.write "<td>" & r.fields("tel") & "</td>"
        response.write "</tr>"
        r.movenext
loop
response.write "</table><hr>"
%>
```

# ממשק מנהל

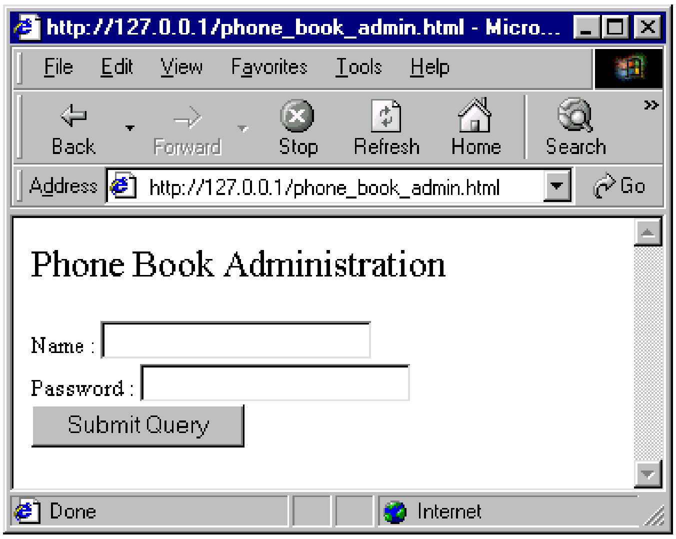
ספר הטלפונים עובד, כולל דפים אישיים ואף חיפוש, וכעת כל שנותר, הוא לבנות ממשק עדכון והוספה למנהלי הספר, כלומר, מסך אדמיניסטרציה.

נתחיל בבניית מסך ההזדהות. מסך זה יכיל טופס וייבנה כ- HTML פשוט.

Phone_book_admin.html

```
<font size=5>Phone Book Administration</font>
<form action=phone_book_admin.asp method=post>
Name : <input type=text name=n><br>
Password : <input type=password name=p><br>
<input type=submit>
</form>
```

כך ייראה מסך זה :




כעת נבנה את הקובץ phone_book_admin.asp המופיע כ- Action של הטופס, ואליו יישלחו הפרמטרים n ו- p.

קובץ זה יבדוק אם שם המשתמש והסיסמה קיימים בטבלת users ובמידה וכן, יאפשר לראות את הנתונים. הבדיקה תתבצע על ידי שאילתה המחפשת שילוב של השם והסיסמה. במידה ואין תשובה לשאילתה (EOF, זוכרים?), השילוב של שם וסיסמה אינם נכונים. שימו לב, כי אנו משתמשים בשיטת Post מטעמי סודיות, ולכן חובה להשתמש ב- request.form. כך תתבצע הבדיקה :

```
set c=server.createobject("adodb.connection")
c.open "dsn=phone;"
set r=server.createobject("adodb.recordset")
r.open "select * from users where user='" & request.form("n") &_
 "' and pass='" &_
request.form("p") & "'"
if r.eof then
   response.write "<font size=5>Access Denied<hr>"
   response.end
```

אם כן, אם הסיסמה אינה נכונה, יש לחסום את גישת המשתמש ולהוסיף response.end כדי להיות בטוחים, שחלקים נוספים מהמסמך לא יישלחו לגולש לא מורשה.

במקרה שהסיסמה נכונה, יש להציג למשתמש המורשה את הנתונים. בספר זה בחרנו להציג למשתמש המורשה את אותה טבלה שרואה הגולש הרגיל, אך לדאוג שהקישורים מהשמות הפרטיים, יובילו למסכי פרטים אישיים בהם יוכל המנהל לבצע שינויים, הוספות או מחיקות.

עלינו לאבטח את שאר המסמכים הנמצאים ביישום, ולכן בחרנו להשתמש בעוגיות. כך שברגע שגולש הזדהה בהצלחה, נשתול לו עוגיית אישור אשר תיבדק בכל דף מאובטח. שימו לב, כי בבניית הקישורים בטבלה, החלפנו את ...<a href=personal_page ב- ...<a href=secure_personal_page. זאת כדי לאפשר למנהל הספר להגיע לדף מאובטח בו יוכל לשנות את הנתונים.

להלן הקוד שכולל בדיקת עוגיה מאובטחת למקרה של גישה חוזרת לדף לאחר הזדהות:

phone_book_admin.asp

```
<%
set c=server.createobject("adodb.connection")
c.open "dsn=phone;"
set r=server.createobject("adodb.recordset")
r.activeconnection=c
r.open "select * from users where user='" & request.form("n") & "' and pass='" &_
request.form("p") & "'"
if r.eof and request.cookies("authorization")<>"yes" then
  response.write "<font size=5>Access Denied<hr>"
  response.end
end if
r.close
 response.cookies("authorization")="yes"
%>
<body>
<font size=5><center>Welcome to my phone book application -
Admin</center></font>
<hr>
<form>
Search for:<br>
First Name:<input type=text name=fn> Last Name <input type=text name=ln><input
type=submit value=Search></form>
<%
if not isempty(request.querystring("fn")) then
 if request.querystring("fn")="" then
  if request.querystring("ln")<>"" then
     type_of_query=" lname='" & request.querystring("ln") & "'"
  end if
 else
  if request.querystring("ln")="" then
     type_of_query=" fname='" & request.querystring("fn") & "'"
  else
     type_of_query=" fname='" & request.querystring("fn") & "' and lname='" &_
```

```
        request.querystring("ln") & ""
    end if
  end if

  if isempty(type_of_query) then
     response.write "<b>You did not type a person name!"
  else
     r.open "select * from personal_details where" & type_of_query
     if r.eof then
        response.write "<b> The person you asked for is not listed!"

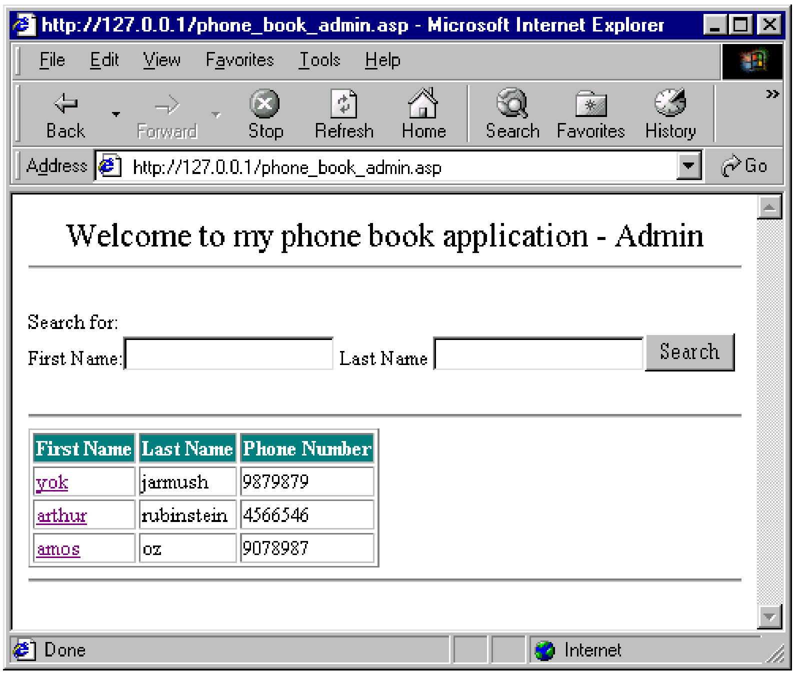
        r.close
     else
        Response.write "First Name : " & r.fields("fname") & "<br>"
        Response.write "Last Name : " & r.fields("lname") & "<br>"
        Response.write "Phone Number : " & r.fields("tel")
        r.close
     end if
  end if
end if

%>
<hr>
<%

r.open "select * from personal_details"

response.write "<table border><tr>"
response.write "<th bgcolor=teal><font color=white>First Name</th>"
response.write "<th bgcolor=teal><font color=white>Last Name</th>"
response.write "<th bgcolor=teal><font color=white>Phone Number</th>"
response.write "</tr>"
do until r.eof
            response.write "<tr>"
            response.write "<td><a href=secure_personal_page.asp?p=" &
            r.fields("fname") &_
             ">" & r.fields("fname") & "</a></td>"
            response.write "<td>" & r.fields("lname") & "</td>"
            response.write "<td>" & r.fields("tel") & "</td>"
            response.write "</tr>"
            r.movenext
loop
response.write "</table><hr>"
%>
```

כך ייראה מסך האדמיניסטרציה:



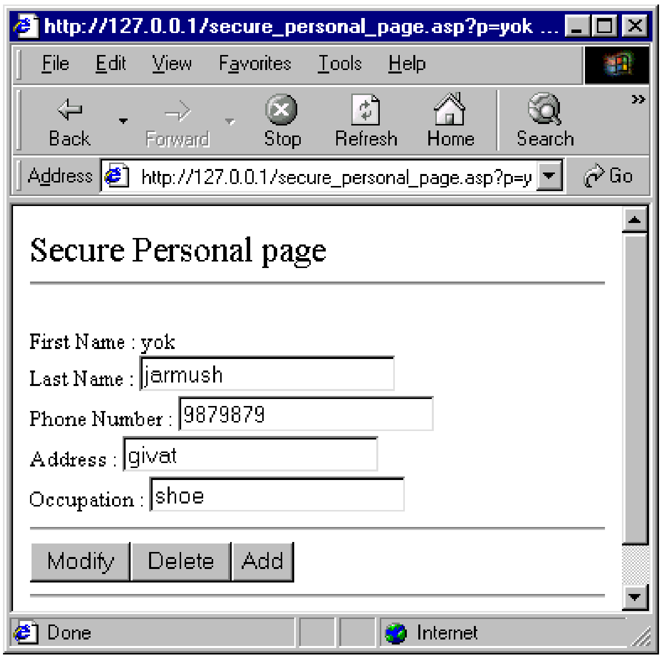

כעת, ניגש לבנות את דף העדכון המאובטח. לפני כל פעילות, יבדוק הדף המאובטח את העוגיה authorization כדי לאמת כי המשתמש הזדהה בצורה מסודרת. לאחר מכן נציג את המידע בתוך שדות טקסט, זאת כדי לאפשר את שינוי הפרטים (מלבד שמו הפרטי של האדם, אשר בשל חוסר במפתח ראשי, מזהה כל אדם בצורה חד ערכית).

כך נבצע את בדיקת העוגיה והצגת הנתונים:

```
<%
if request.cookies("authorization")<>"yes" then
   response.write "<font size=5>Access Denied<hr>"
   response.end
else
set c=server.createobject("adodb.connection")
c.open "dsn=phone;"
set r=server.createobject("adodb.recordset")
r.activeconnection=c
r.open "select * from personal_details where fname='" &_
 request.querystring("p") & "'"
response.write "<font size=5>Secure Personal page</font><hr>"
response.write "<form name=formi>"
response.write "First Name : " & r.fields("fname") & "<br>"
response.write "Last Name : <input name=ln value=" & r.fields("lname") & "><br>"
response.write "Phone Number : <input name=tel value=" & r.fields("tel") & "><br>"
```

```
response.write "Address : <input name=add value=" & r.fields("address") & "><br>"
response.write "Occupation : <input name=oc value=" &_
 r.fields("occupation") & "><hr>"
response.write "<input type=button name=modify value=Modify>"
response.write "<input type=button name=delete value=Delete>"
response.write "<input type=button name=add value=Add><hr>"
response.write "<a href=phone_book_admi.asp>Back to Phone Book</a>"
end if
%>
```

כך ייראה המסך :




בשלב זה, שלושת הלחצנים אינם פעילים. כדי לאפשר ללחצנים לפעול, אין מנוס מכתיבת קוד JavaScript המשנה את ה- Action של הטופס בכל פעם לקובץ ASP שיטפל בהזנה, שינוי או הוספה.

לחילופין, ניתן לפתור בעיה זו על ידי קישור היפר-טקסט פשוט.

להלן פונקציית JavaScript אשר תקבע את ה- Action של הטופס ותשלח אותו :

```
function set_form_action(destination)
{
document.formi.action=destination;
document.formi.submit()
}
```

חשוב לציין כי עלינו לשלוח תמיד גם את שם האדם (מכיון שהוא מספק לנו זיהוי חד ערכי). אין אדם המופיע בשדה ולכן לא יישלח שם, ולכן נוסיף שדה נסתר (Hidden) אשר יכיל את שם האדם באופן הבא:

```
response.write "<input type=hidden name=fn value=" & r.fields("fn") & ">"
```

באופן זה, יישלח שמו הפרטי של האדם עם הטופס.

עוד נושא למחשבה הוא האם להשתמש בתפיסת Reentry, או לבנות קובץ ASP עבור כל פעולה. התשובה היא לעולם הרגשתו האישית של התוכניתן. אם הוא חש כי יוכל לבנות מסמך Reentry מסודר וקל לתחזוקה, ניתן לבנות קובץ אחד. מכיון שמסמך Reentry הינו ארוך ומסובך יחסית, נבנה בדוגמה זו מספר קבצי ASP קטנים.

תחילה, נוסיף את פונקציית JavaScript ואת ההפניות אליה מהלחצנים. להלן הקובץ הגמור:

secure_personal_page.asp

```
<%
if request.cookies("authorization")<>"yes" then
   response.write "<font size=5>Access Denied<hr>"
   response.end
else
set c=server.createobject("adodb.connection")
c.open "dsn=phone;"
set r=server.createobject("adodb.recordset")
r.activeconnection=c
r.open "select * from personal_details where fname='" &_
 request.querystring("p") & "'"
response.write "<font size=5> Secure Personal page</font><hr>"
response.write "<form name=formi>"
response.write "<input type=hidden name=fn value=" &_
 r.fields("fname") & ">"
response.write "First Name : " & r.fields("fname") & "<br>"
response.write "Last Name : <input name=ln value='" &_
 r.fields("lname") & "'><br>"
response.write "Phone Number : <input name=tel value='" &_
 r.fields("tel") & "'><br>"
response.write "Address : <input name=ad value='" &_
 r.fields("address") & "'><br>"
response.write "Occupation : <input name=oc value='" &_
 r.fields("occupation") & "'><hr>"
response.write "<input type=button name=modify value=Modify" &_
"onclick=set_form_action('modify.asp')>"
response.write "<input type=button name=delete value=Delete
onclick=set_form_action('delete.asp')>"
```

```
response.write "<input type=button name=add value=Add
onclick=set_form_action('add.asp')><hr>"
response.write "<a href=phase3.asp>Back to Phone Book</a>"
end if
%>
<script>
function set_form_action(destination)
{
document.formi.action=destination;
document.formi.submit();
}
</script>
```

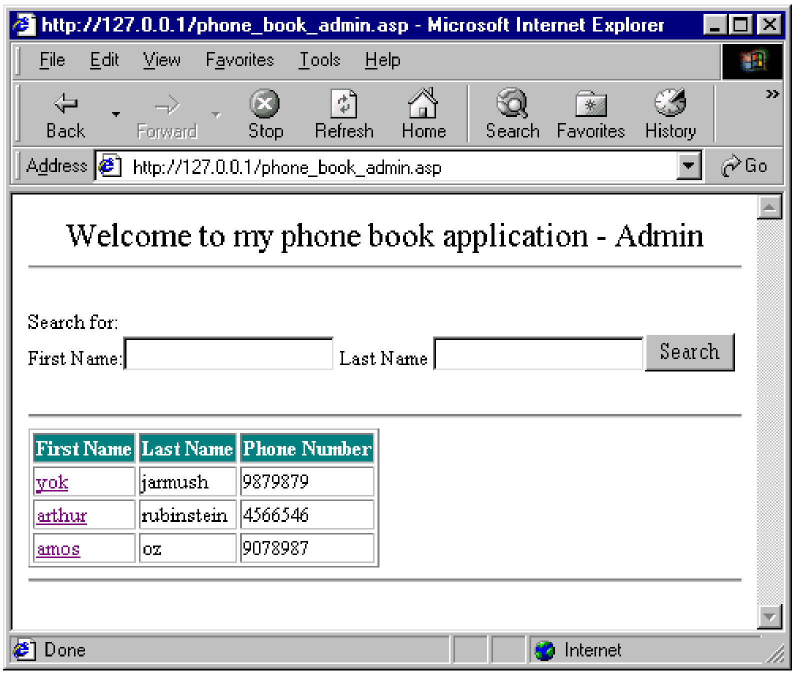
# עדכון

הקובץ הראשון שנבנה הוא הקובץ שיטפל בשינוי הפרטים. על קובץ זה לקבל את הנתונים ולהחליף אותם בטבלת personal_details על פי השם הפרטי. כמובן, שבתור קובץ מאובטח, יש לבצע תחילה בדיקת העוגיה authorization. לאחר סיום הפעילות, יפנה הקובץ את המשתמש ל- phone_book_admin.asp על ידי redirect, כדי לראות את השינוי בטבלה.

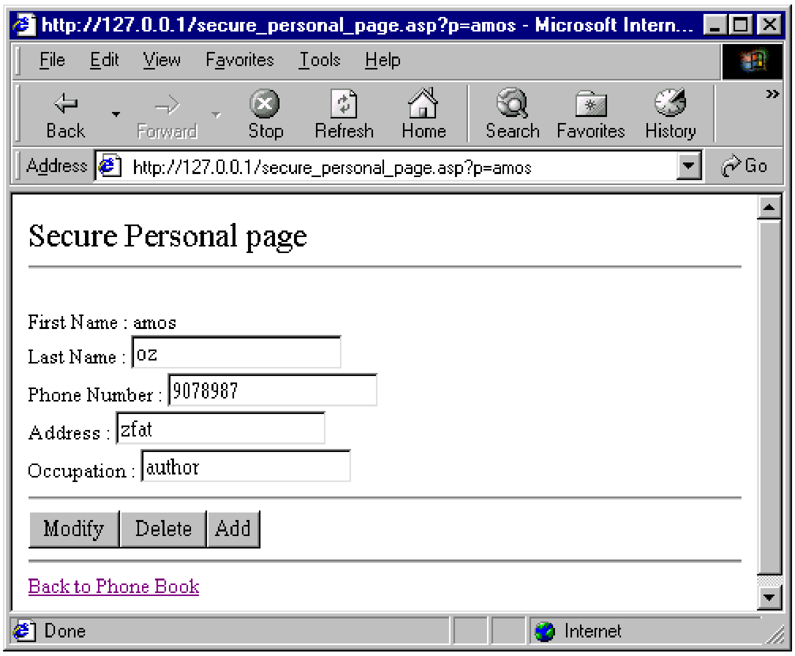
Modify.asp

```
<%
if request.cookies("authorization")<>"yes" then
   response.write "<font size=5>Access Denied<hr>"
   response.end
else
  set c=server.createobject("adodb.connection")
  c.open "dsn=phone;"
  c.execute "update personal_details set lname='" &_
 request.querystring("ln") & "', tel='" &_
  request.querystring("tel") & "', address='" & request.querystring("ad") &_
"', occupation='" & request.querystring("oc") & "' where fname='" &_
request.querystring("fn") & "'"
c.close
set c=nothing
response.redirect "phone_book_admin.asp"
end if
%>
```

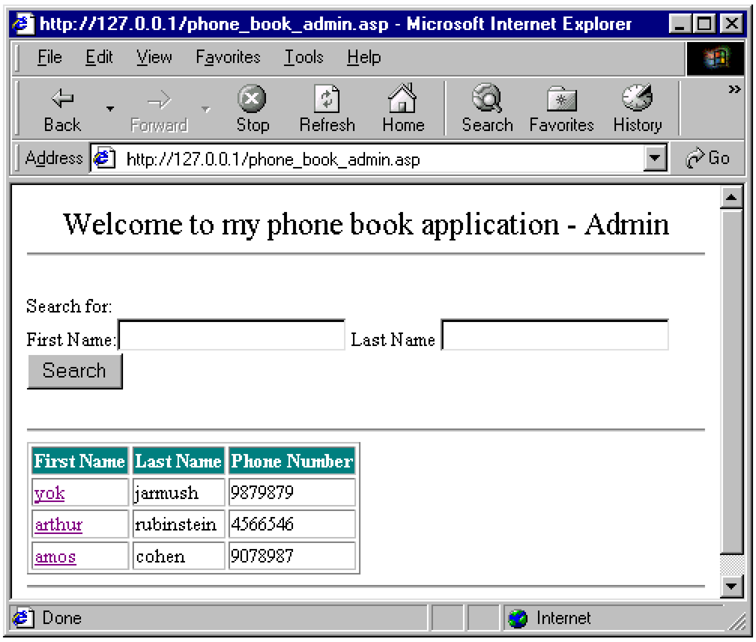
הבא נבחן את היישום עד כה. וזהו מסך האדמיניסטרציה:




אם נרצה לשנות את פרטיו האישיים של amos, נלחץ על שמו הפרטי ונקבל את המסך הבא:



נשנה את שם משפחתו ל- cohen ונלחץ על לחצן **Modify**.




ניתן לראות בטבלה, כי השינוי במסד הנתונים התבצע.

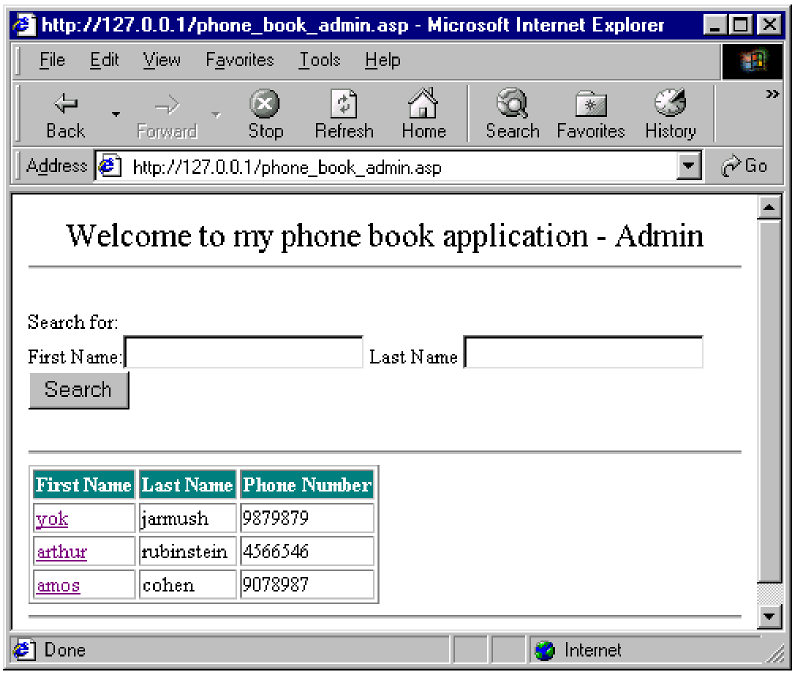
# מחיקה

פעולת מחיקה היא פעולה פשוטה. כל שעלינו לעשות הוא למחוק את הרשומה לפי השם הפרטי באופן הבא:

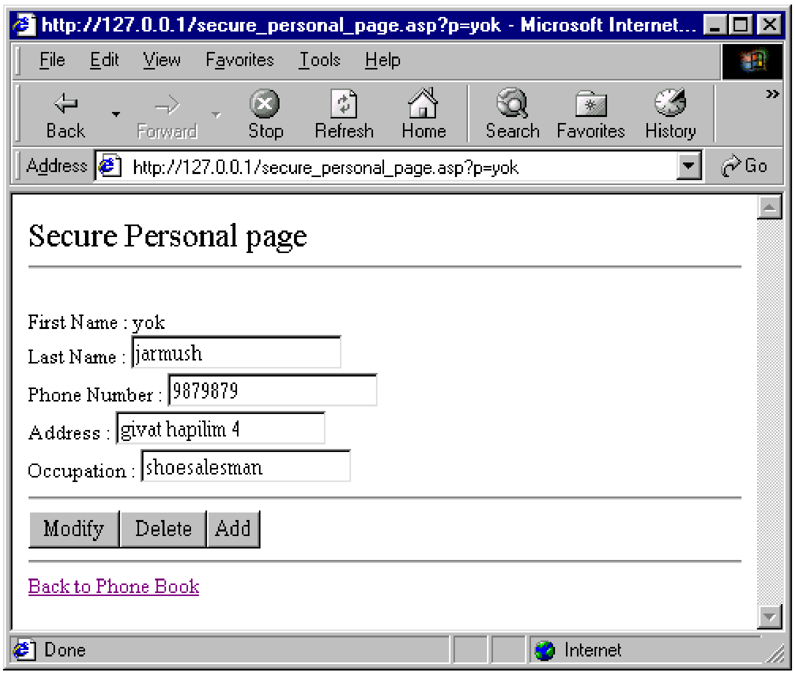
delete.asp

```
<%
if request.cookies("authorization")<>"yes" then
   response.write "<font size=5>Access Denied<hr>"
   response.end
else
set c=server.createobject("adodb.connection")
  c.open "dsn=phone;"
  c.execute "delete from personal_details where fname='" &_
   request.querystring("fn") & "'"
  c.close
  set c=nothing
  response.redirect "phone_book_admin.asp"
end if
%>
```

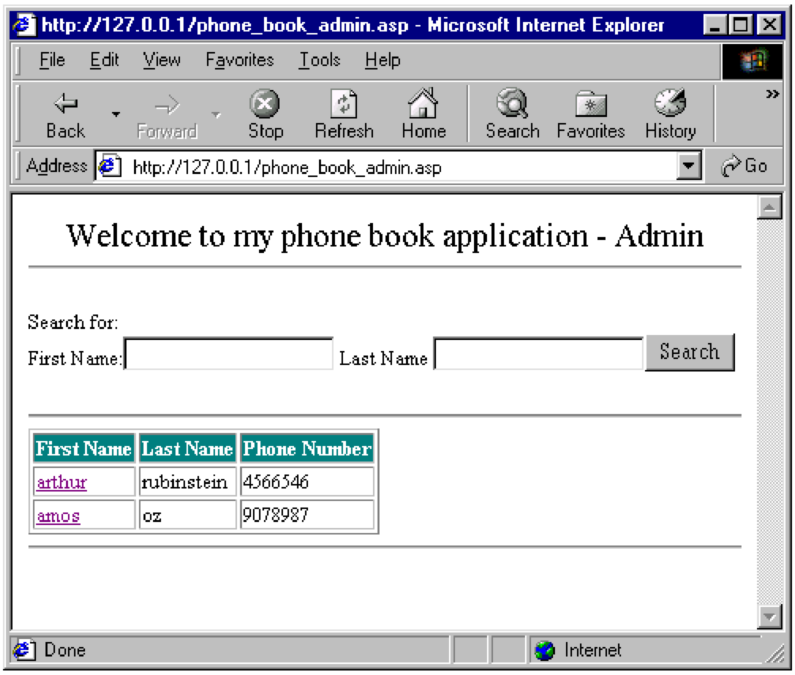
כעת נבצע תהליך מחיקה.




במסך האדמיניסטרציה, נבחר את שם האדם אותו אנו מעוניינים למחוק, ונגיע לדף הפרטים האישיים המובאים.



כל שנותר, הוא ללחוץ על לחצן **Delete** ולקבל את המסך הראשון.




כפי שניתן לראות, הרשומה המבוקשת נמחקה ממסד הנתונים.

# הוספה

מסך ההוספה אינו מסובך במיוחד, אך מתחלק לשני חלקים. החלק הראשון, הוא הצגת שדות טקסט ריקים שיאפשרו הזנת פרטי האדם החדש. החלק השני הוא הזנת המידע אל מסד הנתונים. מכיון שפעולות אלו אינן מסובכות מדי, נבצע את שתיהן בקובץ ASP אחד בתפיסת Reentry.

בכל מקרה, מכיון שאנו עוסקים בדפים מאובטחים, נבדוק תמיד כי העוגיה authorization קיימת. לאחר מכן נבדוק אם הדף עולה בפעם הראשונה על ידי בדיקת isempty לאחד השדות ונציג את המידע בהתאם.

החלק הראשון ייראה כך:

```
if request.cookies("authorization")<>"yes" then
    response.write "<font size=5>Access Denied<hr>
    response.end
end if
if isempty(request.querystring("first_name") then
  response.write "<font size=5>Add a new person</font><hr>"
  response.write "<form>"
```

```
  response.write "First Name : <input name=first_name><br>"
  response.write "Last Name : <input name=last_name><br>"
  response.write "Phone Number : <input name=phone_number><br>"
  response.write "Address : <input name=address><br>"
response.write "Occupation : <input name=occupation><hr>"
response.write "<input type=submit value=Submit><hr></form>"
```

אם התנאי :

```
if isempty(request.querystring("first_name") then
```

אינו מתקיים, יבצע קובץ זה פעולת בדיקה ראשונית לוודא כי השם הפרטי אינו קיים בטבלה (להזכירכם, אנו זקוקים לשם פרטי ייחודי מכיון שאין לנו **מפתח ראשי**). אם השם אינו קיים, יוזנו הפרטים לטבלת personal_details במסד הנתונים.

להלן הקוד המלא :

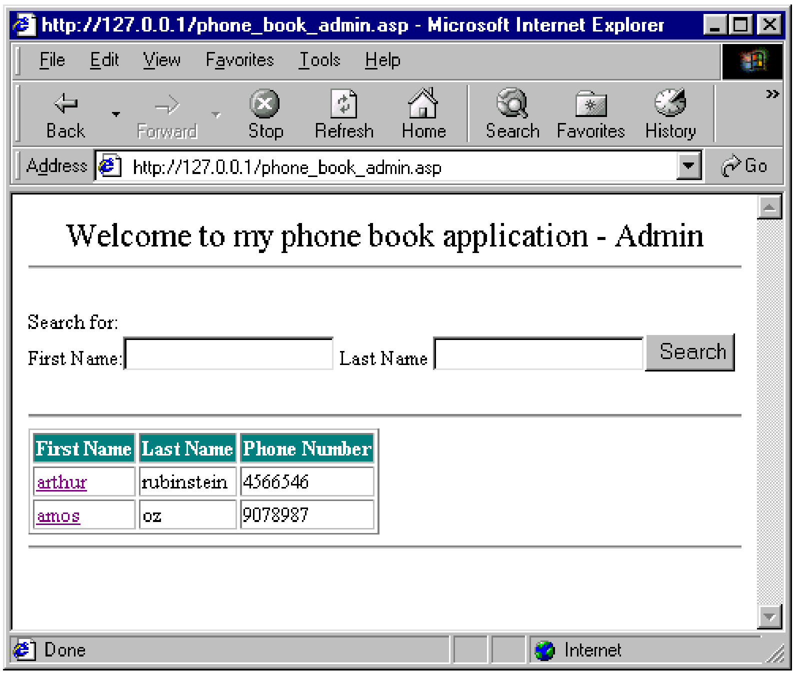
add.asp

```
<%
if request.cookies("authorization")<>"yes" then
    response.write "<font size=5>Access Denied<hr>"
    response.end
end if
if isempty(request.querystring("first_name")) then
  response.write "<font size=5>Add a new person</font><hr>"
  response.write "<form>"
  response.write "First Name : <input name=first_name><br>"
  response.write "Last Name : <input name=last_name><br>"
  response.write "Phone Number : <input name=phone_number><br>"
  response.write "Address : <input name=address><br>"
  response.write "Occupation : <input name=occupation><hr>"
  response.write "<input type=submit value=Submit><hr></form>"
else
 set c=server.createobject("adodb.connection")
 c.open "dsn=phone;"
 set r=server.createobject("adodb.recordset")
r.activeconnection=c
 r.open "select fname from personal_details where fname='" &_
 request.querystring("first_name") & "'"
   if r.eof then

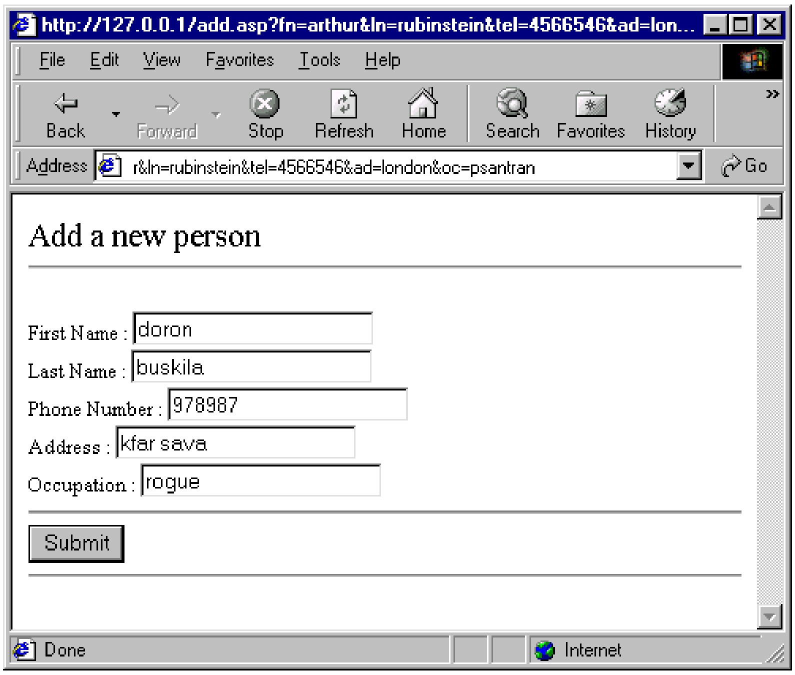
      c.execute "insert into personal_details values('" &_
      request.querystring("first_name") & "','" &_
      request.querystring("last_name") & "','" &_
      request.querystring("phone_number") & "','" &_
```

```
      request.querystring("address") & "','" &_
      request.querystring("occupation") & "')"
      c.close
      set c=nothing
      response.redirect "phone_book_admin.asp"
    else
      response.write "The First Name you chose already exists.<br>"
      response.write "<a href=add.asp>Back</a>"
    end if
end if
%>
```

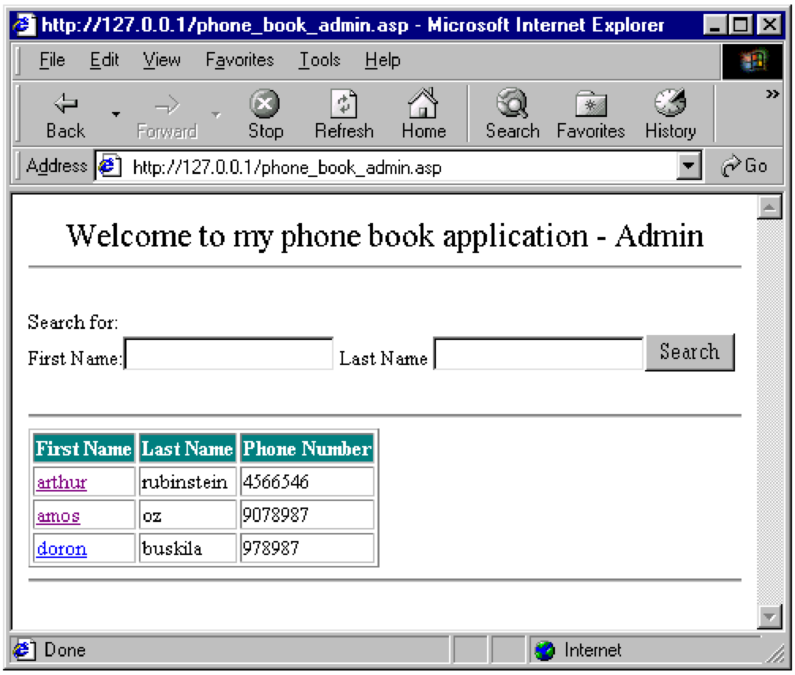
כעת נבחן את הוספת הרשומה. נפתח את מסך האדמיניסטרציה:




נלחץ על אחת הרשומות, ולאחר מכן על לחצן **Add** ונמלא את השדות.




כל שנותר הוא ללחוץ על לחצן **Submit** ולבחון את התוצאה:



שיפורים אפשריים למערכת יהיו לחצן Log out אשר יבטל את העוגיה ויחזיר את הגולש למצב משתמש רגיל. כדאי גם להעביר את לחצן Add למסך הראשי המציג את הטבלה ועוד.

יישום זה מהווה בסיס טוב להמשך פיתוח יישומים מבוססי מסדי נתונים כגון חנות וירטואלית, פורום קבוצת דיון וכד׳.

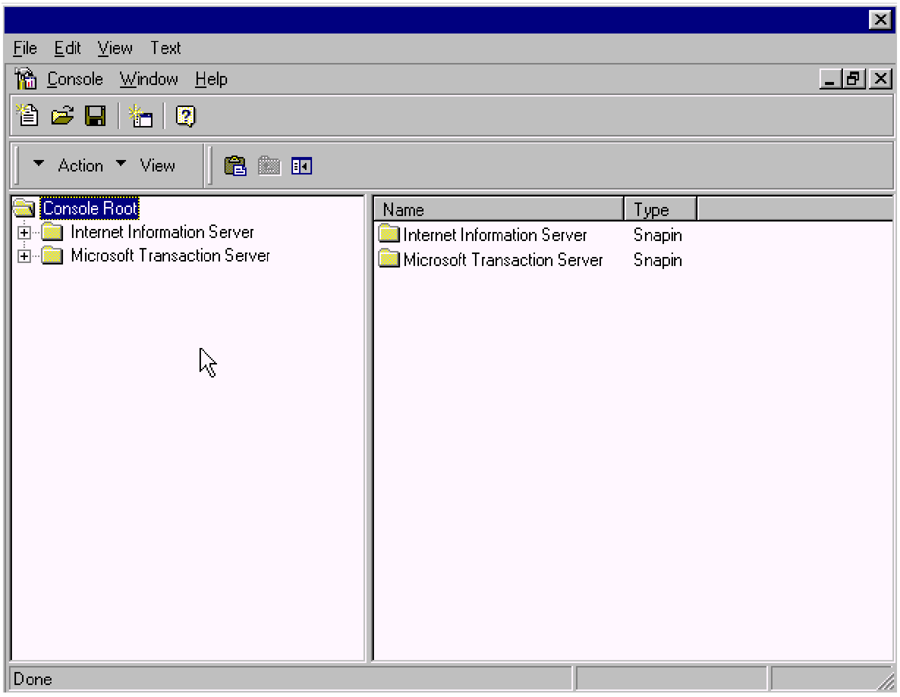
היישום המלא נמצא בתיקיה phone_book_application בתקליטור המצורף. כל שנחוץ כדי להריצו הוא להעתיק את תוכן התיקיה הזו אל תיקיית inetpub\wwwroot במחשב שלכם (במערכת הפעלה מסוג NT, winnt\inetpub\wwwroot) לאחר התקנת השרת, ולהגדיר את הקובץ phone_book.mdb ב- ODBC כ- system DSN מסוג MS Access Driver בשם phone, ולגשת מהדפדפן אל:

127.0.0.1/phone_book_application.asp

## פרק 12

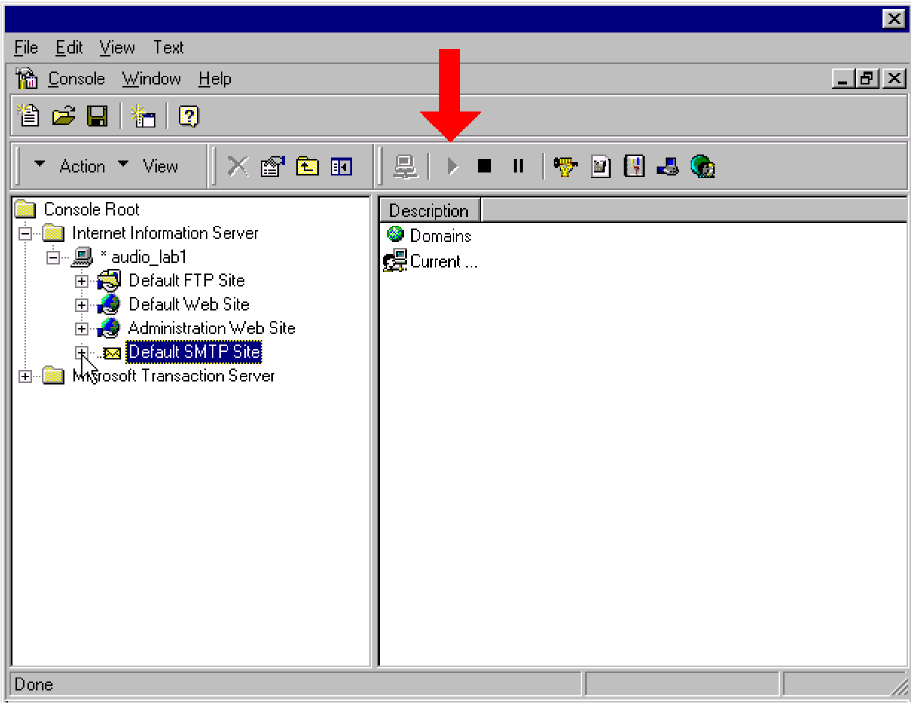
# שילוב הודעות דואר אלקטרוני

כדי לייצר ולשלוח הודעות דואר אלקטרוני יש צורך בשרת NT בעל שירות SMTP פעיל.




כדי לוודא כי שירות SMTP פעיל, יש להיכנס לחלון הניהול של IIS (Microsoft Management Console).

כעת יש להגיע לשירות SMTP באופן הבא:




אם לחצן ההפעלה (Play) המסומן בחץ הוא אפור, השירות פעיל. אם הוא שחור, יש ללחוץ עליו. כעת, יש לייצר את אובייקט הדואר. אובייקט הדואר נוצר על פי האובייקט cdonts.newmail (Collaboration Data Object NT Server), באופן הבא:

```
set m=server.createobject("cdonts.newmail")
```

לאחר שייצרנו את אובייקט הדואר **m**, ניתן לקבוע מספר שדות דואר אלקטרוני סטנדרטיים כגון:

m.from="joe@get.com" → קביעת כתובת השולח

m.to="all@tree.com" → קביעת כתובת הנמען

m.cc="sub@junga.net" → קביעת כתובת המכותבים לידיעה

m.bcc="group@svensk.org" → קביעת כתובת המכותבים הנסתרים לידיעה

m.attachefile="c:\kooki.txt" → קביעת הקובץ המצורף

m.attachURL="http://www.godzila.com" → קביעת כתובת מצורפת

m.bodyFormat=1 → קביעת פורמט ההודעה כטקסט(1) או כ- אתר(0)

m.subject="Welcome" → קביעת נושא ההודעה

```
m.body="Welcome new users, How are you today" &_
    chr(13) & "I have great news for you." &_
    chr(13) & "I am comming back soon!"
```

שימו לב, כי בתכונת body הקובעת את גוף ההודעה, משורשרים chr(13). זו למעשה השיטה לשבור שורות בהודעה. Chr(13) מייצג את Carriage return השובר שורה ומחזיר את הסמן לתחילת השורה הבאה.

כעת, כל שנותר הוא לשלוח את ההודעה על ידי:

```
m.send
```

אם ברשותכם שרת NT בעל שירות SMTP פעיל, תוכלו לבנות יישומים כגון רשימות תפוצה מבוססות מסד נתונים או קובץ טקסט, וכן לדאוג כי בכל אירוע מסוים או תקופת זמן קבועה מראש, ישלח האתר הודעת דואר אלקטרוני אליכם.

פרק 13

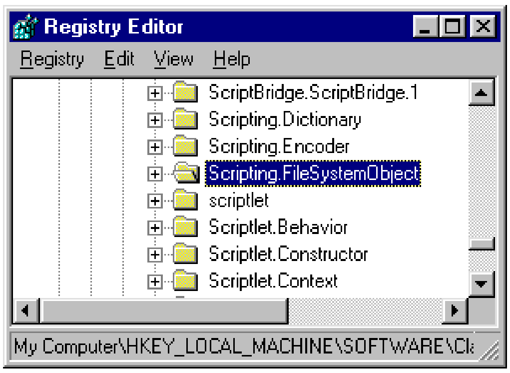
# עבודה עם קבצים

אחת האפשרויות הנפתחות בפני מפתחי ASP היא הטיפול בקבצים על השרת. ניתן ליצור קבצים, לקרוא ולכתוב אליהם ואף למחוק. היישומים לאפשרויות אלו הם רבים. מכיון שקיימת שליטה על תהליך ייצור הקבצים, ניתן לייצר כל פורמט הנוח למנהלים ומתחזקי האתר לצרכי רישום ומעקב. ניתן גם לייצר פורמטים הנוחים לקריאה על ידי גופי צד שלישי אשר לעיתים מעדיפים לעבוד מול קבצי טקסט שטוחים (Flat files), מאשר מול מסדי נתונים מבוססי טבלאות.

## יצירת קובץ

כדי לטפל בקובץ יש להשתמש באובייקט הלוגי File. תהליך הייצור של האובייקט File אינו פשוט, מכיון שאת אובייקט File ניתן לייצר רק על ידי שימוש באובייקט FileSystemObject, אשר הינו אובייקט מערכת.

לכן נפעל בשני שלבים. בשלב הראשון נייצר אובייקט FileSystemObject על ידי שימוש בתחביר המוכר לנו מייצור אובייקטים, כגון connection ו-recordset. מכיון ש-FileSystemObject רשום במערכת כך:




נשתמש בשיטה createobject של אובייקט server באופן הבא:

```
set fs=server.createobject("scripting.filesystemobject")
```

השלב השני, יהיה להשתמש באובייקט filesystemobject כדי לייצר את אובייקט file לו אנו זקוקים. לשם כך נשתמש בשיטה createfile של אובייקט filesystemobject באופן הבא (ניצור את **f** כאובייקט מסוג file):

```
set f=fs.createtextfile("c:\kooki.txt")
```

שורת קוד זו תייצר את הקובץ kooki.txt בספריה \:c. אם נרצה לרשום טקסט אל הקובץ, נוכל כבר להשתמש באובייקט file שלנו (**f**), על ידי התכונה write.

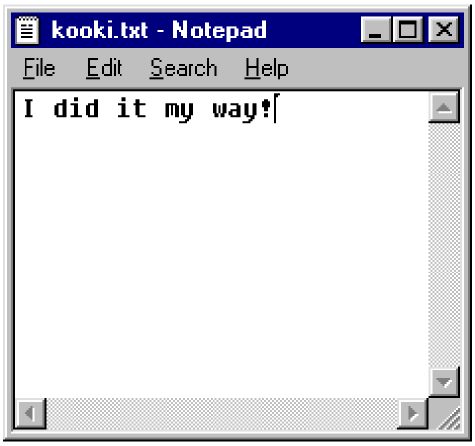
```
f.write "How do you do?"
```

זהו הקוד המלא ליצירת קובץ טקסט וכתיבה לתוכו:

create_text_file.asp

```
<%
set fs=server.createobject("scripting.filesystemobject")
set f=fs.createtextfile("c:\kooki.txt")
f.write "I did it my way!"
%>
Check out c:\kooki.txt
```

אם אין הגבלת הרשאות במערכת ההפעלה שלכם (במקרה של שרת מסודר), תוכלו לראות כי הקובץ נוצר והמלל "I did it my way!" נכתב לתוכו.




אם נריץ את create_text_file.asp בשנית, יידרס הקובץ kooki.txt הישן על ידי החדש. ניתן למנוע מצב זה על ידי הוספת המילה false באופן הבא:

```
f=fs.createtextfile("c:\kooki.txt",false)
```

# קריאת קובץ קיים

לפתיחת קובץ קיים נשתמש בשיטת openTextfile של אובייקט file. שיטה זו מקבלת שלושה פרמטרים.

הפרמטר הראשון הוא שם הקובץ הרצוי.

הפרמטר השני הוא לאיזו מטרה נפתח הקובץ כאשר:

1 – מסמן פתיחה לקריאה בלבד

2 – מסמן פתיחה לכתיבה (במקום הטקסט הקיים)

8 – מסמן פתיחה למטרת הוספה

הפרמטר השלישי קובע באיזה פורמט ייפתח קובץ הטקסט:

(0) – מסמן כי הקובץ ייפתח כקובץ ASCII

(-1) – מסמן כי הקובץ ייפתח בפורמט Unicode

(-2) – מסמן כי הקובץ ייפתח על פי ברירת המחדל של המערכת

אם כך, נייצר תחילה אובייקט fileSystemObject:

```
set fs=server.createobject("scripting.filesystemobject")
```

נייצר את אובייקט file כ- **f** ונשתמש בשיטה openTextFile כדי לפתוח את הקובץ kooki.txt אשר יצרנו בדוגמה הקודמת:

```
set f=fs.opentextfile("c:\kooki.txt",1,0)
```

לאחר מכן נציג את תוכן הקובץ תוך שימוש בשיטה חדשה של אובייקט file: readall

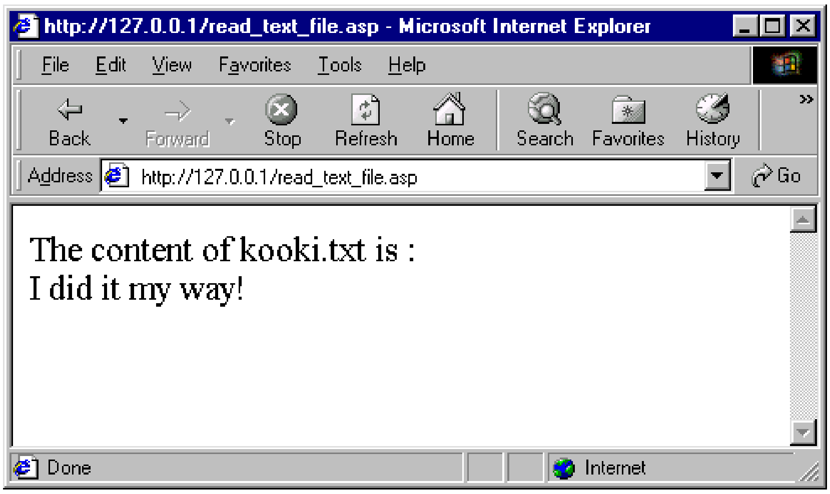
```
response.write "<font size=5>The content of kooki.txt is :<br>" & f.readall
```

להלן הקוד המלא:

read_text_file.asp

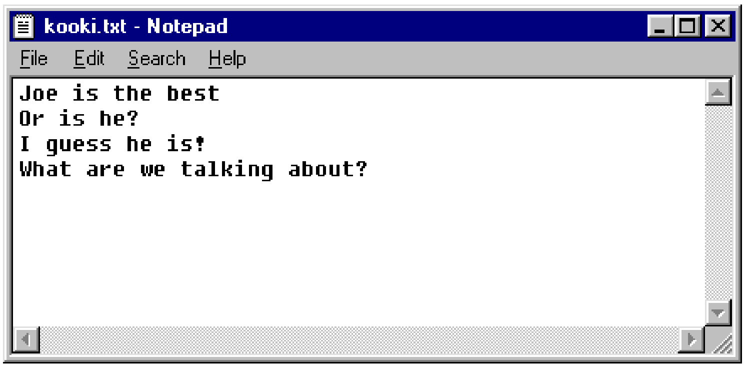
```
<%
set fs=server.createobject("scripting.filesystemobject")
set f=fs.opentextfile("c:\kooki.txt",1,0)
response.write "<font size=5>The content of kooki.txt is :<br>" & f.readall
%>
```

להלן התוצאה:




## טיפול במספר שורות רב

כאשר קוראים מקובץ בעזרת השיטה openTextFile, נמצא הסמן הלוגי (Index) בתחילת השורה הראשונה בקובץ. מבנה זה מזכיר את מבנה אובייקט Recordset. כדי לקרוא שורה אחת בלבד נשתמש בשיטה readline במקום השיטה readall. השיטה readline תקרא שורה אחת בלבד ותעביר את Index לשורה הבאה. נפתח את הקובץ kooki.txt בפנקס הרשימות, נוסיף לו מספר שורות ונשמור אותו באופן הבא:




כעת ניצור את אובייקט fileSystemObject הבסיסי:

set fs=server.createobject("scripting.filesystemobject")

נשתמש בשיטת opentextfile של כדי לייצר את **f** כאובייקט מסוג file:

set f=fs.opentextfile("c:\mooki",1,0)

כעת נבקש לקרוא את השורה הראשונה ולשלוח אותה לגולש:

```
response.write "The first line is: " & f.readline & "<br>"
```

לצורך הבנה, נציג גם את השורה השנייה. להזכירכם, עם סיום קריאת השורה הראשונה, עובר Index לשורה הבאה ולכן ניתן לקרוא אותה:

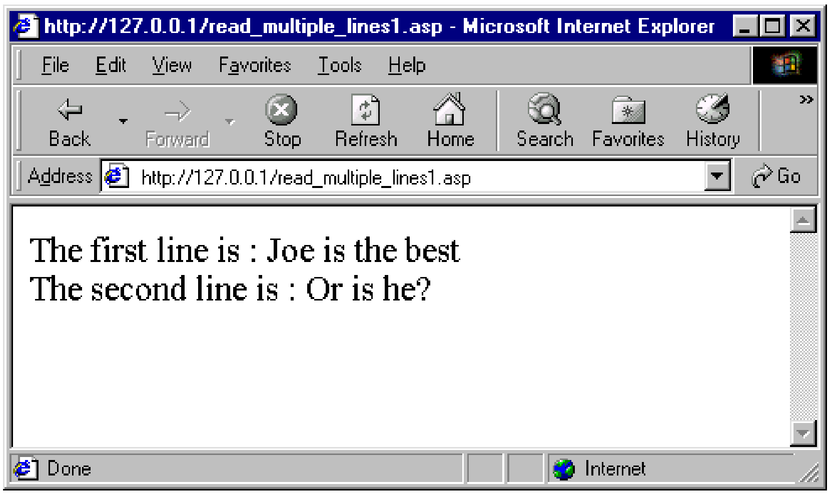
```
response.write "The second line is: " & f.readline & "<br>"
```

להלן הקוד המלא:

read_multiple_lines1.asp

```
<%
set fs=server.createobject("scripting.filesystemobject")
set f=fs.opentextfile("c:\kooki.txt",1,0)
response.write "<font size=5>"
response.write "The first line is : " & f.readline & "<br>"
response.write "The second line is : " & f.readline & "<br>"
%>
```

כך תיראה התוצאה:



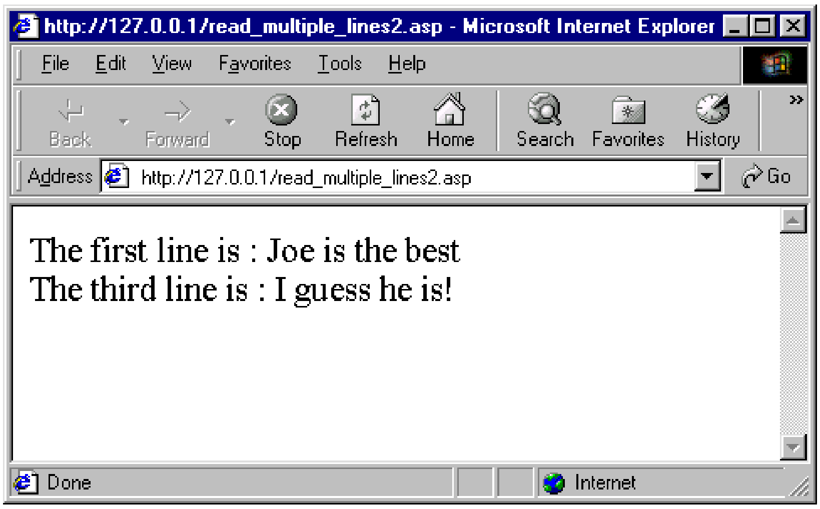

כדי לדלג על שורה, נוכל להשתמש בשיטה skipline של אובייקט file. בקוד הבא נרצה להציג את השורה השלישית ולכן לאחר קריאת השורה הראשונה נבצע skipline ונקרא מייד את השורה השלישית באופן הבא:

read_multiple_lines2.asp

```
<%
set fs=server.createobject("scripting.filesystemobject")
set f=fs.opentextfile("c:\kooki.txt",1,0)
```

```
response.write "<font size=5>"
response.write "The first line is : " & f.readline & "<br>"
f.skipline
response.write "The third line is : " & f.readline
%>
```

ניתן לראות כי אכן התבצע דילוג מעל לשורה השנייה:




אם נרצה לעבור על כל השורות במסמך טקסט, נוכל להשתמש בתכונה atEndOfStream המייצגת את סוף הקובץ, באופן המזכיר ריצה על אובייקט Recordset:

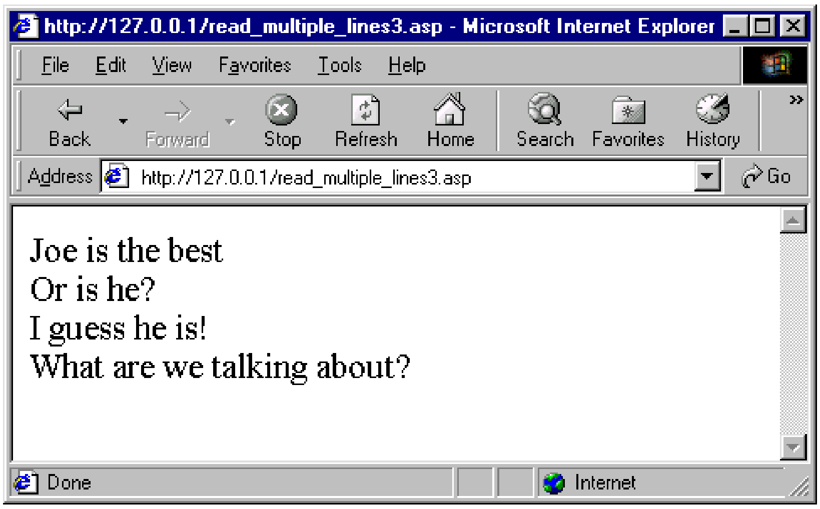
```
do until f.atendofstream
response.write f.readline
loop
```

כלומר,

read_multiple_lines3.asp

```
<%
set fs=server.createobject("scripting.filesystemobject")
set f=fs.opentextfile("c:\kooki.txt",1,0)
response.write "<font size=5>"
do until f.atendofstream
        response.write f.readline & "<br>"
loop
%>
```

להלן התוצאה :




# מחיקת קובץ

כדי למחוק קובץ יש לייצר אובייקט fileSystemObject :

```
set fs=server.createobject("scripting.filesystemobject")
```

לאחר מכן, יש לייצר אובייקט file תוך שימוש בשיטה getFile באופן הבא :

```
set f=fs.getfile("c:\kooki.txt")
```

בשלב זה ניתן להשתמש בשיטה Delete של האובייקט **f** (אובייקט מסוג file) :

```
f.delete
```

להלן קוד המוחק את הקובץ kooki.txt :

delete_text_file.asp

```
<%
set fs=server.createobject("scripting.filesystemobject")
set f=fs.getfile("c:\kooki.txt")
f.delete
%>
File deleted
```

# עוד טיפול בקבצים

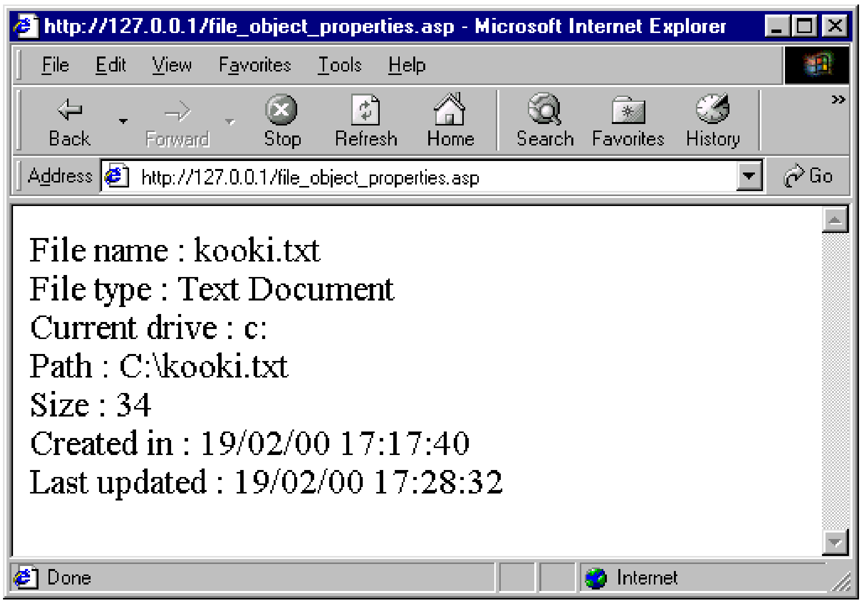
ייצור אובייקט file בעזרת opentextfile או createtextfile לא מאפשר גישה לתכונות הקובץ. כדי לייצר אובייקט file עם גישה לתכונותיו, יש להשתמש בשיטה getFile. רק שיטה זו מאפשרת ייצור של אובייקט file באופן בו ניתן לגשת אל תכונותיו הרבות.

הקוד הבא מדגים הצגה של מספר תכונות של קובץ קיים :

file_object_properties.asp

```
<%
set fs=server.createobject("scripting.filesystemobject")
set f=fs.getFile("c:\kooki.txt")
response.write "<font size=5>"
response.write "File name : " & f.name & "<br>"
response.write "File type : " & f.type & "<br>"
response.write "Current drive : " & f.drive & "<br>"
response.write "Path : " & f.path & "<br>"
response.write "Size : " & f.size & "<br>"
response.write "Created in : " & f.datecreated & "<br>"
response.write "Last updated : " & f.datelastmodified & "<br>"
%>
```

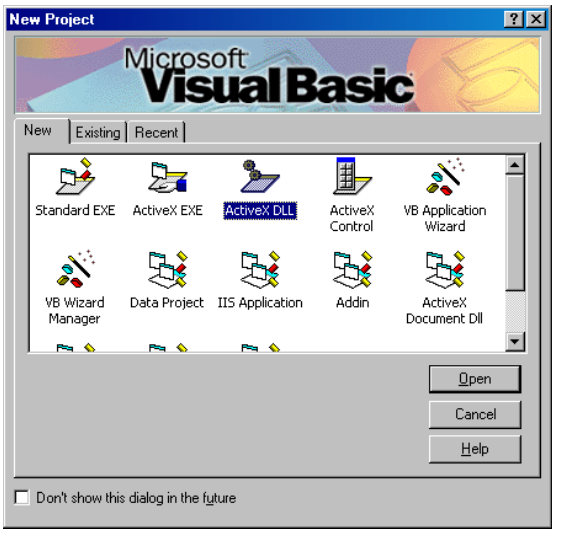
להלן התוצאה :



## פרק 14

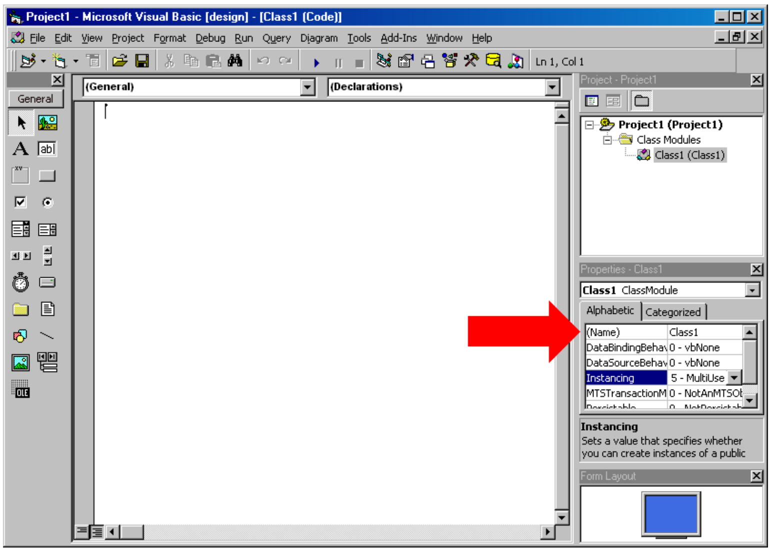
# יצירת אובייקטי COM באמצעות Visual Basic 6

פרק זה מיועד למתכנתים ב-Visual Basic. כאן נלמד לייצר אובייקטים כגון האובייקטים המלווים ספר זה (adodb.connection, scripting.filesystemobject וכד׳). ההיגיון מאחורי בניית אובייקטים הוא רב. החל בעובדה פשוטה כי קוד ASP הוא איטי בשל הדרך בה הוא נקרא ומתבצע מקבצי טקסט ואילו אובייקטים מהודרים לשפת מכונה מספקים מהירות וביצועים עדיפים, וכלה ביתרונות התפיסה המכוונת עצמים (קיצור קוד, שימוש חוזר ועוד). בשל העובדה שהאובייקטים נבנים על פי סטנדרט COM (Component Object Model) הנתמך בשפות רבות, ניתן לבנות את אותם אובייקטים ב- C/C++, Java, Visual Basic ועוד. בספר זה נבחן את שפת Visual Basic בשל פשטותה. עם פתיחת סביבת העבודה של Visual Basic 6, נקבל את המסך הבא:

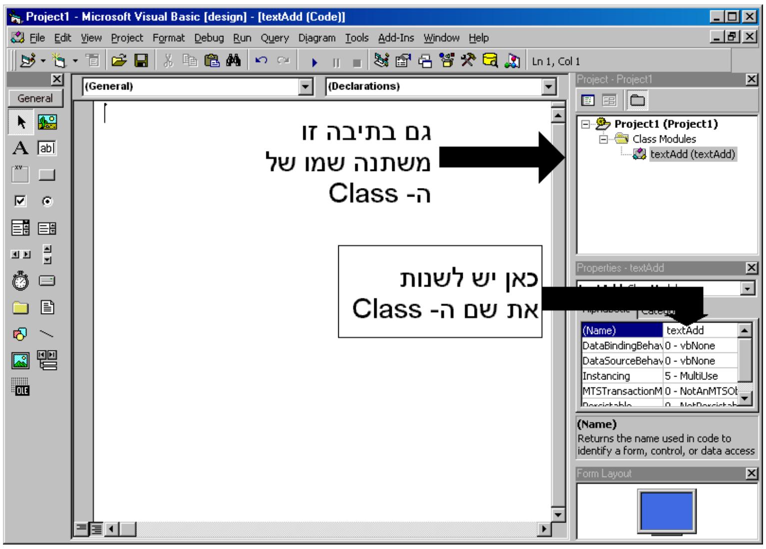



במסך זה יש לבחור ActiveX DLL. למעשה, ניתן ליישם אובייקט גם כ- ActiveX EXE אך כיון שאובייקט מבוסס EXE מתבצע כתהליך נפרד, נפגמים ביצועי המערכת. יתרונו היחסי של EXE הוא העובדה שגם אם תהליך EXE נופל, לא ייפגע שאר היישום. ברוב

המקרים, ישנה העדפה להשתמש ב-DLL בשל התחשבות בביצועים. לאחר בחירת ActiveX DLL, נקבל את המסך הבא:




שימו לב לחץ. חץ זה מצביע על תיבת התכונות ובתוכה על תכונת השם. כיון שבחלון העליון המסומן Class1, תיבת התכונות מכילה את תכונותיו, נשנה את שמו של Class1 ל- textAdd באופן הבא:




שימו לב, כי גם בתיבה העליונה (המסומנת בחץ), השתנה שמו של Class ל-textAdd. אם כך, אנו עומדים ליצור Class (למעשה תבנית לאובייקט) שיאפשר לשרשר טקסט למחרוזות קיימות ולכן שמו הוא textAdd. שמו של הפרויקט הוא Project1. ניתן לשנות זאת על ידי סימונו בתיבה העליונה ושינוי השם בתיבת התכונות התחתונה, אך

בשלב זה נשאיר את שם הפרויקט Project1. למעשה, כשם שאובייקט מסוג connection מתבסס על התבנית adodb.connection יתבססו אובייקטים מסוג textAdd על התבנית Project1.textAdd. בשלב הבא כאשר נאתחל את משתנה X כאובייקט מסוג textAdd, נשתמש בתחביר:

```
Set x=server.createobject("project1.textAdd")
```

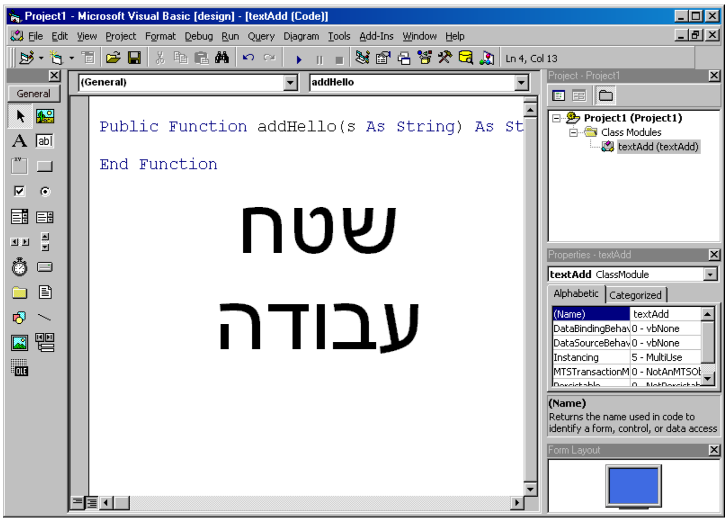
מטרת האובייקט החדש שניצור, תהיה לשרשר את המילה Hello לכל מחרוזת שתישלח אליו. כדי לאפשר שליחת נתונים והפעלה ניאלץ ליצור שיטה ל-Class שיצרנו. לשיטה נקרא addHello והתחביר בו נשתמש יהיה זה:

```
Set x=server.createobject("Project1.textAdd")
Response.write x.addHello("mooshon")
```

השיטה addHello של האובייקט x (מסוג textAdd), תשרשר למחרוזת שקיבלה כפרמטר ("mooshon") את המחרוזת "Hello" והתוצאה על מסך הגולש תהיה:

Hello Mooshon

אם כך, עלינו לבנות את השיטה addHello. נתחיל ונכתוב את התחביר בשטח העבודה של Visual Basic:




התחביר לבניית שיטה הוא כדלהלן:

```
Public function addHello(s as string) as string
```

כפי שניתן לראות, השיטה (function) addHello, מקבלת פרמטר אחד מסוג מחרוזת (String), ומכנה אותו בשם **s**. לאחר הסוגריים, ניתן לראות את הביטוי as string אשר משמעותו היא כי השיטה (function) גם תחזיר מחרוזת כיוון שכזכור, על השיטה להחזיר את המחרוזת שקיבלה בתוספת המחרוזת "Hello". התחביר לקביעת תוכן המחרוזת שתחזיר השיטה הוא:

```
addHello="hello" & s
```

כלומר, השיטה addHello תחזיר את המחרוזת "Hello" משורשרת למחרוזת **s** שנשלחה על ידי יוזם הקריאה לשיטה. כעת, נסגור את השיטה על ידי התחביר:

```
End function
```

למעשה, ברגע שנסיים להקליד את שורת פתיחת השיטה, תוסיף סביבת העבודה של Visual Basic את הסגירה. כך תיראה השיטה המלאה:

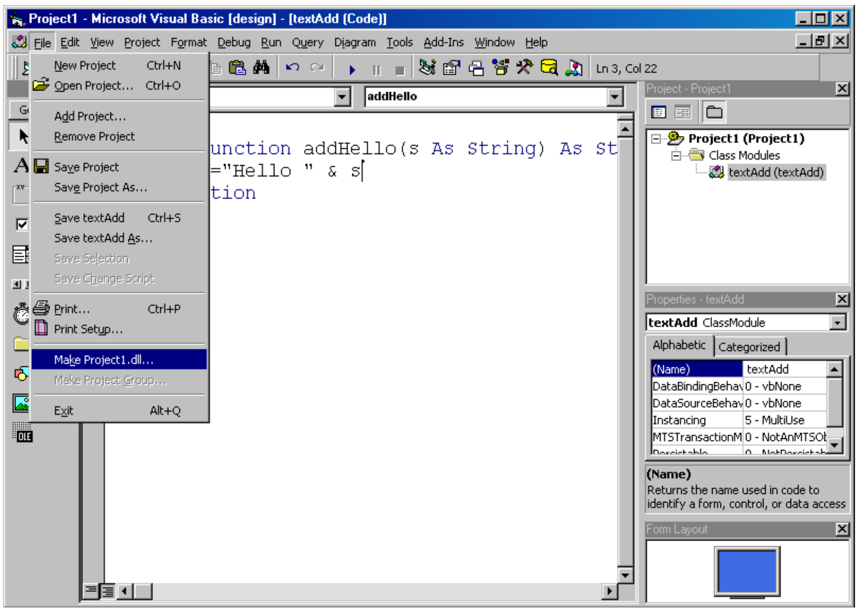
```
Public function addHello(s as string) as string
addHello="hello" & s
End function
```

## רישום ה- Class

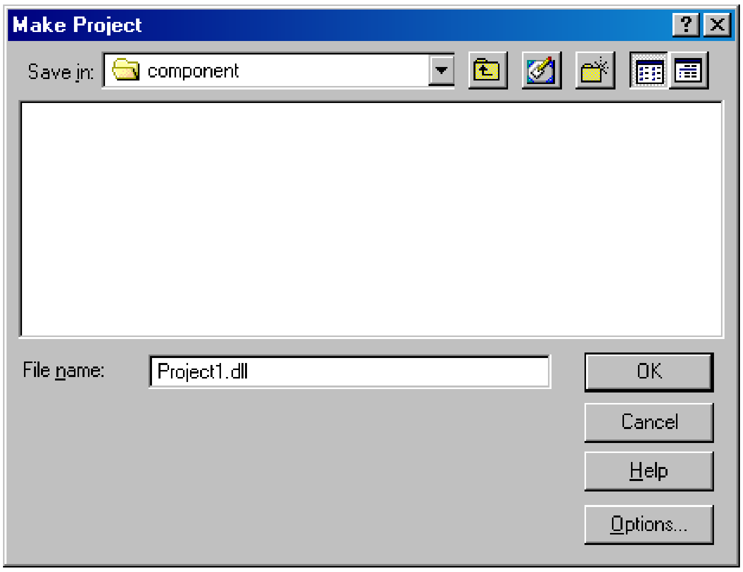
השימוש במילה אובייקט שגוי, כיון שאובייקט הינו המשתנה הנוצר מתבנית Class הרשום במערכת. כלומר אם נרשום:

```
Set x=server.createobject("adodb.connection")
```

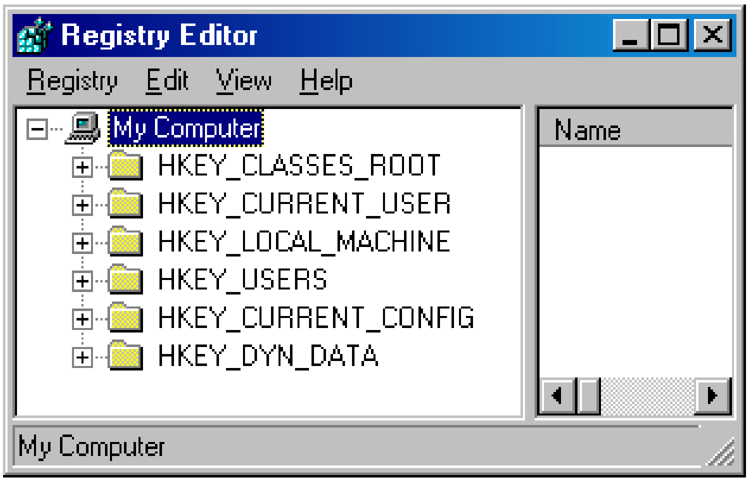
יהווה x אובייקט, בעוד ש- Connection הוא למעשה Class או התבנית שעל פיה נוצר האובייקט x. אם כן, כדי לפנות ל-Class שיצרנו מתוך מסמך ASP, עלינו לדאוג כי הוא יופיע ברישום המערכת (Registry), כשם שכל שאר ה-Classes בהם השתמשנו במהלך הספר מופיעים. לצורך כך נבחר בתפריט **File** ← **Make Project1.DLL** באופן הבא:



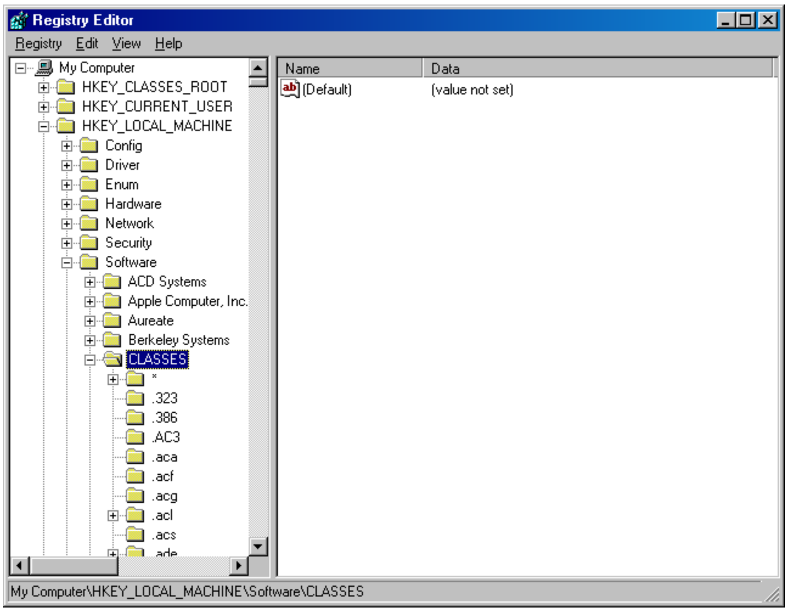
שמרו את הקובץ Project1.dll בתיקיה מסוימת באופן הבא:



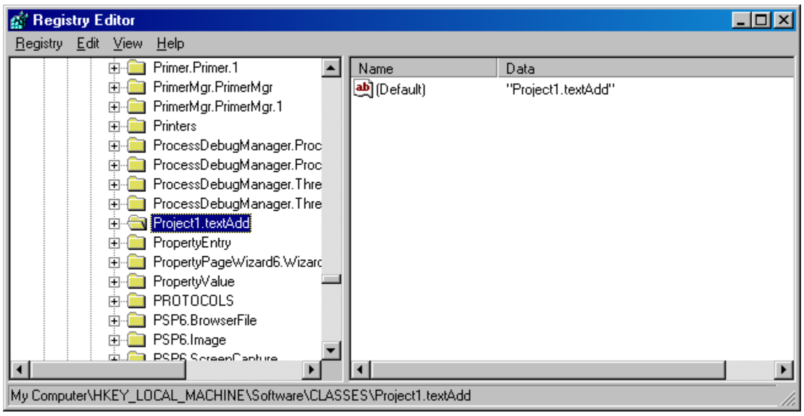

כעת, רושמת סביבת העבודה של Visual Basic את DLL במערכת המקומית. כדי לבדוק כי הרישום התבצע כהלכה, נלחץ על **התחל←הפעלה** (Start>Run) ונקיש את המילה **regedit**. זהו המסך שנקבל:



נפתח את Hkey_Local_Machine ולאחריו את Software ולסיום את Classes.




בתוך Classes נאתר את Project1.textAdd




אם Project1.textAdd אינו מופיע, או במידה וקיים צורך להפנות ל-Class על מחשב שאינו המחשב עליו נמצא ASP, יש לכתוב בשורת הפקודה (Run) :

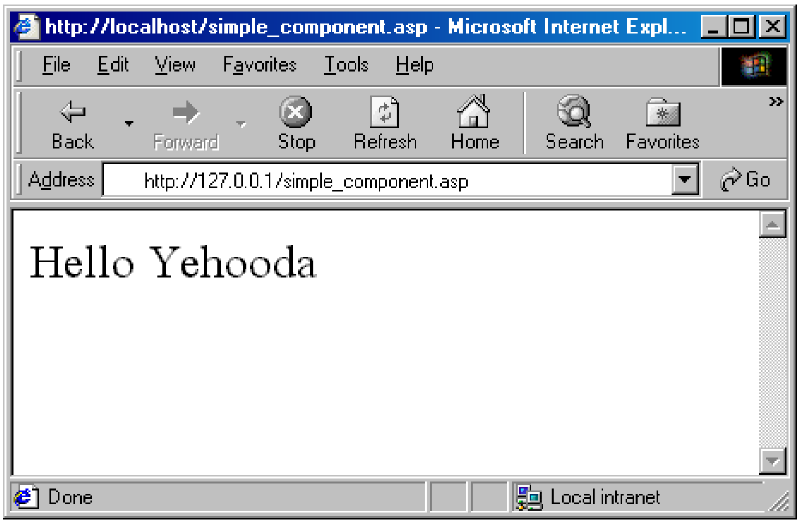
```
regsvr32 c:\components\Project1
```

בכל מקרה, ברגע ש-Class רשום במערכת, ניתן לגשת אליו מקוד ASP. נכתוב את הקוד הפשוט הבא:

Simple_component.asp

```
<%
set x=server.createobject("Project1.textAdd")
response.write x.addHello("Yehooda")
%>
```

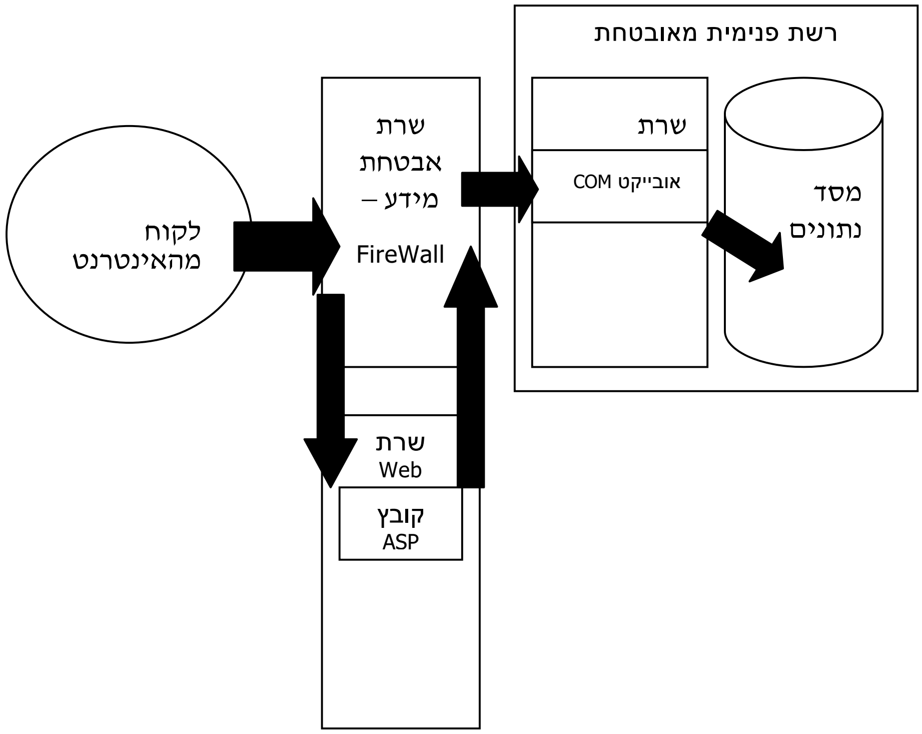
זו תהיה התוצאה:




# יצירת יישום מבוסס מסד נתונים באמצעות אובייקטים

היישום המשמעותי ביותר של אובייקטי COM הוא בעבודה מול מסדי נתונים. כיון שאובייקט COM גנוז מקוד מהודר בשפת מכונה, רמת אבטחת המידע שבו גבוהה עשרות מונים מזו של קובץ ASP המכיל טקסט פשוט. כמו כן שימוש באובייקטי COM משפר את ביצועי המערכת.

להלן ארכיטקטורה לוגית עבור מערכת אופיינית.

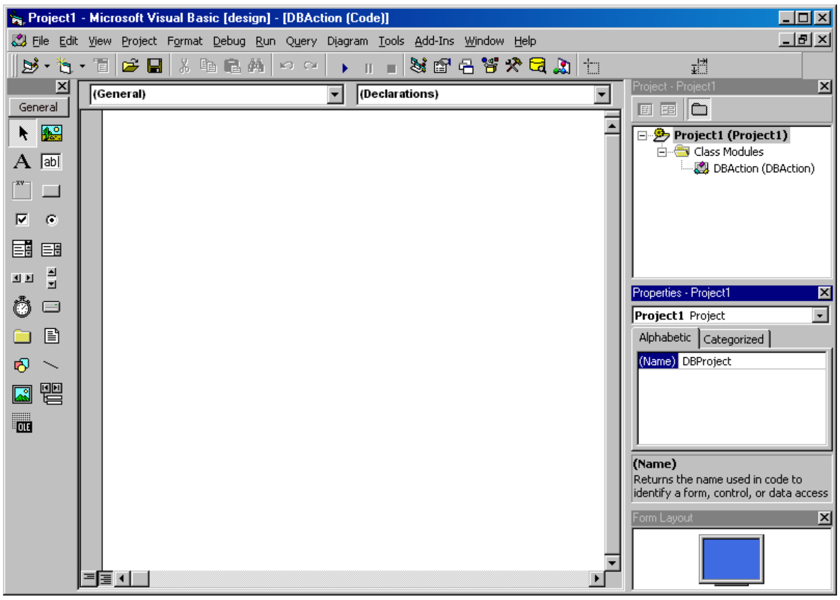



לפי הארכיטקטורה הנ"ל, מסד הנתונים נמצא ברשת פנימית מאובטחת ביחד עם רשת היישומים (Application Server) המכיל את האובייקטים בעלי גישה לנתונים. אובייקטים אלו מכילים את ההיגיון העסקי (Business Logic). כיון שאין לרשת זו גישה מהאינטרנט, רמת האבטחה שבה גבוהה. לעומת זאת, כיון שיש צורך לתת לגולשים אפשרות להגיע לשרת Web, מוקמת רשת חצי מאובטחת המכונה **DMZ** (Demiliterized Zone) ומקימים בה שרת Web. שרת Web אינו מוגן לחלוטין אך גם אינו פרוץ לחלוטין. בכל מקרה, על גבי שרת Web ממוקמים רק קבצי ASP המהווים ממשק פשוט. כיון שאין גישה למסדי הנתונים דרך ASP, גם לו ייפרץ השרת Web, לא יסוכנו הנתונים. עבור כל שליפה או עדכון נתון ממסד הנתונים, פונה ASP לאובייקט COM הממוקם ברשת המאובטחת, וזה בתורו פונה למסד הנתונים.

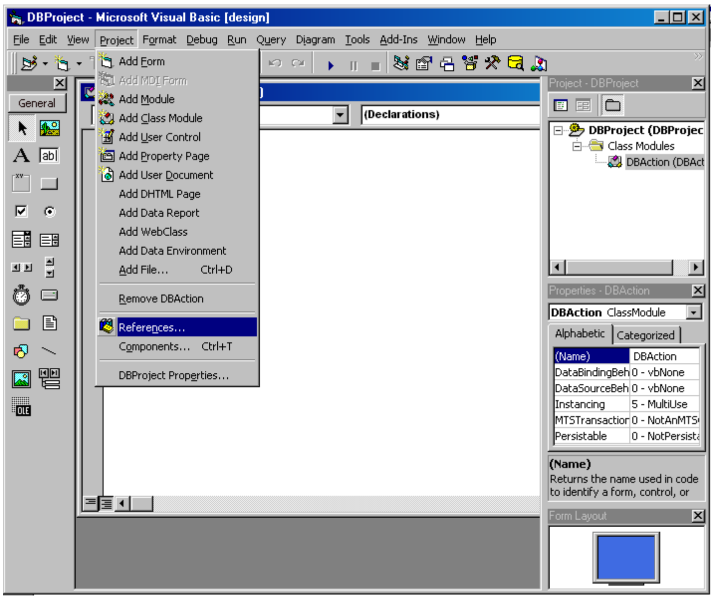
ה-Class הבא שניצור, יהיה Class המאפשר גישה למסד נתונים. למעשה ניצור יישום המוודא שם וסיסמה. דבר זה יתבצע על ידי אובייקט COM ולא על ידי דף ASP. היתרון הבטיחותי הוא שגם אם ייפרץ השרת וייפתחו קבצי ASP, כל שיוכלו לראות יהיה אתחול אובייקט COM, במקום לראות משפטי SQL המבהירים את מבנה מסד הנתונים ודרך הגישה אליו. לצורך כך, נשתמש בטבלת Users המוגדרת בטבלה phone_book.mdb (טבלת המשתמשים ביישום phone book application). אלו מכם אשר אינם משתמשים ב-Access, יוכלו ליצור טבלה תוך שימוש ב- Microsoft Text

Driver כפי שנלמד בפרק 11, יישומים מבוססים מסדי נתונים. על הטבלה להכיל עמודת טקסט בשם name ועמודת טקסט בשם pass. כמו כן, אם אינכם משתמשים בטבלה הקיימת, עליכם להזין לפחות שם וסיסמה אחת ולהגדיר DSN ב-ODBC.

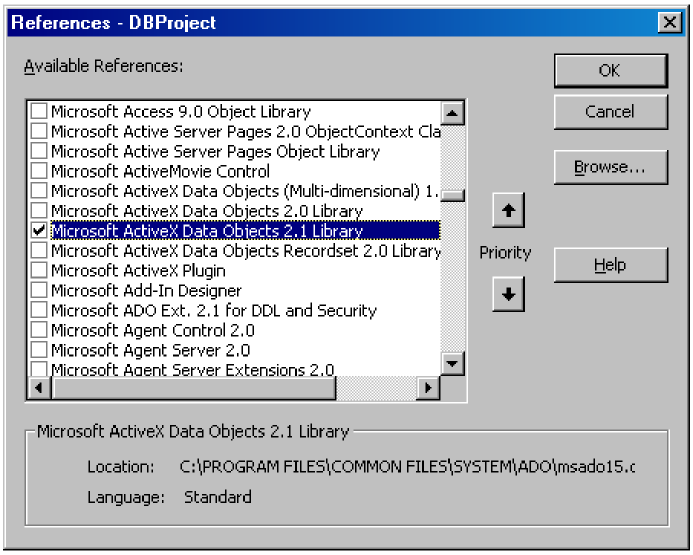
נפתח את סביבת העבודה של Visual Basic 6 ונבחר שוב ActiveX DLL. נקרא ל- Class בשם DBAction ולפרויקט בשם DBProject באופן הבא:




כעת נבחר בתפריט **project**>**reference**.



ולפנינו ייפתח המסך הבא :




מסך זה מאפשר לכלול ספריות אובייקטים חיצוניות לקוד. כיון שאנו מעוניינים לפנות למסד נתונים, עלינו להשתמש באובייקטים של מודל ADO. כדי לפנות לאובייקטים החיצוניים הללו עלינו לסמן את תיקיית Microsoft ActiveX Data Objects. בצילום מעל נבחרה תיקיה מגרסה 2.1. אם במחשב בו אתם בונים את היישום קיימת גירסה נמוכה יותר, מומלץ להוריד את הגירסה העדכנית מאתר הבית של מיקרוסופט. לאחר הוספת ההתייחסות לתיקיית ADO, יש להתחיל ולכתוב שיטה אשר תקבל את שם המשתמש והסיסמה מדף ASP ותבצע בדיקה מול מסד הנתונים. נתחיל בפתיחת השיטה באמצעות התחביר הבא :

```
Public function verifyPassword(user as string,pass as string) as string
```

כפי שניתן לראות, השיטה תקבל שני משתנים מסוג מחרוזת ותחזיר משתנה אחד מסוג מחרוזת. כעת נכריז על אובייקט מסוג connection. שימו לב להבדל בין תחביר ASP לתחביר Visual Basic :

```
Dim myCon As New ADODB.Connection
myCon.Open "DSN=phone"
```

ההכרזה על recordset תיראה כך :

```
Dim myRec As New ADODB.Recordset
myRec.ActiveConnection = myCon
myRec.Open "select * from users where name='" & CStr(user) &_
 "' and pass='" & CStr(pass) & "'"
```

שימו לב ל-Casting המתבצע על המשתנים user ו-pass. תהליך זה מוודא כי המשתנים עוברים כמחרוזות בשל בעיות הנוצרות במערכות מסוימות בהעברת משתנים דרך

recordset. כעת נבצע את הבדיקה הלוגית שתציין כי השאילתה למסד הנתונים הביאה תוצאה או לא:

```
If myRec.EOF Then
	verifyPassword = "OK"
Else
	verifyPassword = "NO"
End If
```

אם השאילתה החזירה תוצאה ממסד הנתונים, תחזיר השיטה verifyPassword את הערך "OK". במידה ואין התאמת שם וסיסמה, כלומר recordset נמצא ב-EOF, תחזיר השיטה את המחרוזת "NO". לאחר מכן נסגור את אובייקט Recordset:

```
myRec.Close
Set myRec = Nothing
End Function
```

להלן הקוד המלא של Class:

```
Public function verifyPassword(user as string,pass as string) as string
Dim myCon As New ADODB.Connection
myCon.Open "DSN=phone"
Dim myRec As New ADODB.Recordset
myRec.ActiveConnection = myCon
myRec.Open "select * from users where name='" & CStr(user) &_
 "' and pass='" & CStr(pass) & "'"
If myRec.EOF Then
	verifyPassword = "NO"
Else
	verifyPassword = "OK"
End If
myRec.Close
Set myRec = Nothing
End Function
```

כל שנותר כעת הוא לייצר DLL מ-Class ולכתוב את קובץ asp שיאתחל את האובייקט DBProject.DBAction וישלח את שם המשתמש והסיסמה שהקליד הלקוח אל השיטה verifyPassword. להלן קוד ASP:

Component_generator.asp

```
<%
if isempty(request.form("u")) then
response.write "<form method=post>"
response.write "User Name:<input type=text name=u><br>"
response.write "Password: <input type=text name=p><br>"
response.write "<input type=submit></form>"
else
```

```
set x=server.createobject("DBProject.DBAction")
if x.verifyPassword(request.form("u"),request.form("p"))="OK" then
response.write "<font size=5>Welcome Authorized User!<hr>"
else
response.write "<font size=5>Access Denied!<hr>"
end if
end if
%>
```

שימו לב כי גם לו נפרץ ASP הנ"ל, אין לפורץ שום מידע על סוג מסד הנתונים, שם וסיסמת מסד הנתונים או מבנהו. כל הלוגיקה של שליפת הנתונים נמצאת ב-Class המהודר.

**הערה**: כל הרכיבים המרכיבים את היישום הנ"ל נמצאים בתיקיה component generation בתקליטור המצורף.

# עברית

בשל העובדה כי הדפדפנים תוכננו לתמוך באנגלית במקור, נוצרה בעיה חמורה בניסיון לכתוב עברית. הדפדפנים החלו כתומכים בפורמט ASCII המגדיר מספר עבור כל אות. להלן טבלת ASCII בסיסית:

| **מספר עשרוני** | **בסיס 8** | **בסיס 16** | **בינארי** | **ייצוג** | |
|---|---|---|---|---|---|
| 000 | 000 | 000 | 00000000 | NUL | (Null char.) |
| 001 | 001 | 001 | 00000001 | SOH | (Start of Header) |
| 002 | 002 | 002 | 00000010 | STX | (Start of Text) |
| 003 | 003 | 003 | 00000011 | ETX | (End of Text) |
| 004 | 004 | 004 | 00000100 | EOT | (End of Transmission) |
| 005 | 005 | 005 | 00000101 | ENQ | (Enquiry) |
| 006 | 006 | 006 | 00000110 | ACK | (Acknowledgment) |
| 007 | 007 | 007 | 00000111 | BEL | (Bell) |
| 008 | 010 | 008 | 00001000 | BS | (Backspace) |
| 009 | 011 | 009 | 00001001 | HT | (Horizontal Tab) |
| 010 | 012 | 00A | 00001010 | LF | (Line Feed) |
| 011 | 013 | 00B | 00001011 | VT | (Vertical Tab) |
| 012 | 014 | 00C | 00001100 | FF | (Form Feed) |
| 013 | 015 | 00D | 00001101 | CR | (Carriage Return) |
| 014 | 016 | 00E | 00001110 | SO | |
| 015 | 017 | 00F | 00001111 | SI | |
| 016 | 020 | 010 | 00010000 | DLE | (Data Link Escape) |
| 017 | 021 | 011 | 00010001 | DC1 (XON) | (Device Control 1) |
| 018 | 022 | 012 | 00010010 | DC2 | (Device Control 2) |
| 019 | 023 | 013 | 00010011 | DC3 (XOFF) | (Device Control 3) |
| 020 | 024 | 014 | 00010100 | DC4 | (Device Control 4) |

| מספר עשרוני | בסיס 8 | בסיס 16 | בינארי | ייצוג | |
|---|---|---|---|---|---|
| 021 | 025 | 015 | 00010101 | NAK | (Negative Acknowledgement) |
| 022 | 026 | 016 | 00010110 | SYN | (Synchronous Idle) |
| 023 | 027 | 017 | 00010111 | ETB | (End of Trans. Block) |
| 024 | 030 | 018 | 00011000 | CAN | (Cancel) |
| 025 | 031 | 019 | 00011001 | EM | |
| 026 | 032 | 01A | 00011010 | SUB | |
| 027 | 033 | 01B | 00011011 | ESC | (Escape) |
| 028 | 034 | 01C | 00011100 | FS | (File Separator) |
| 029 | 035 | 01D | 00011101 | GS | |
| 030 | 036 | 01E | 00011110 | RS | (Request to Send) |
| 031 | 037 | 01F | 00011111 | US | |
| 032 | 040 | 020 | 00100000 | SP | (Space) |
| 033 | 041 | 021 | 00100001 | ! | |
| 034 | 042 | 022 | 00100010 | " | |
| 035 | 043 | 023 | 00100011 | # | |
| 036 | 044 | 024 | 00100100 | $ | |
| 037 | 045 | 025 | 00100101 | % | |
| 038 | 046 | 026 | 00100110 | & | |
| 039 | 047 | 027 | 00100111 | ' | |
| 040 | 050 | 028 | 00101000 | ( | |
| 041 | 051 | 029 | 00101001 | ) | |
| 042 | 052 | 02A | 00101010 | * | |
| 043 | 053 | 02B | 00101011 | + | |
| 044 | 054 | 02C | 00101100 | , | |
| 045 | 055 | 02D | 00101101 | - | |
| 046 | 056 | 02E | 00101110 | . | |
| 047 | 057 | 02F | 00101111 | / | |
| 048 | 060 | 030 | 00110000 | 0 | |
| 049 | 061 | 031 | 00110001 | 1 | |
| 050 | 062 | 032 | 00110010 | 2 | |
| 051 | 063 | 033 | 00110011 | 3 | |
| 052 | 064 | 034 | 00110100 | 4 | |
| 053 | 065 | 035 | 00110101 | 5 | |
| 054 | 066 | 036 | 00110110 | 6 | |
| 055 | 067 | 037 | 00110111 | 7 | |
| 056 | 070 | 038 | 00111000 | 8 | |
| 057 | 071 | 039 | 00111001 | 9 | |
| 058 | 072 | 03A | 00111010 | : | |
| 059 | 073 | 03B | 00111011 | ; | |
| 060 | 074 | 03C | 00111100 | < | |
| 061 | 075 | 03D | 00111101 | = | |
| 062 | 076 | 03E | 00111110 | > | |

| מספר עשרוני | בסיס 8 | בסיס 16 | בינארי | ייצוג |
|---|---|---|---|---|
| 063 | 077 | 03F | 00111111 | ? |
| 064 | 100 | 040 | 01000000 | @ |
| 065 | 101 | 041 | 01000001 | A |
| 066 | 102 | 042 | 01000010 | B |
| 067 | 103 | 043 | 01000011 | C |
| 068 | 104 | 044 | 01000100 | D |
| 069 | 105 | 045 | 01000101 | E |
| 070 | 106 | 046 | 01000110 | F |
| 071 | 107 | 047 | 01000111 | G |
| 072 | 110 | 048 | 01001000 | H |
| 073 | 111 | 049 | 01001001 | I |
| 074 | 112 | 04A | 01001010 | J |
| 075 | 113 | 04B | 01001011 | K |
| 076 | 114 | 04C | 01001100 | L |
| 077 | 115 | 04D | 01001101 | M |
| 078 | 116 | 04E | 01001110 | N |
| 079 | 117 | 04F | 01001111 | O |
| 080 | 120 | 050 | 01010000 | P |
| 081 | 121 | 051 | 01010001 | Q |
| 082 | 122 | 052 | 01010010 | R |
| 083 | 123 | 053 | 01010011 | S |
| 084 | 124 | 054 | 01010100 | T |
| 085 | 125 | 055 | 01010101 | U |
| 086 | 126 | 056 | 01010110 | V |
| 087 | 127 | 057 | 01011111 | W |
| 088 | 130 | 058 | 01011000 | X |
| 089 | 131 | 059 | 01011001 | Y |
| 090 | 132 | 05A | 01011010 | Z |
| 091 | 133 | 05B | 01011011 | [ |
| 092 | 134 | 05C | 01011100 | \ |
| 093 | 135 | 05D | 01011101 | ] |
| 094 | 136 | 05E | 01011110 | ^ |
| 095 | 137 | 05F | 01011111 | _ |
| 096 | 140 | 060 | 01100000 | ` |
| 097 | 141 | 061 | 01100001 | a |
| 098 | 142 | 062 | 01100010 | b |
| 099 | 143 | 063 | 01100011 | c |
| 100 | 144 | 064 | 01100100 | d |
| 101 | 145 | 065 | 01100101 | e |
| 102 | 146 | 066 | 01100110 | f |
| 103 | 147 | 067 | 01100111 | g |
| 104 | 150 | 068 | 01101000 | h |

| מספר עשרוני | בסיס 8 | בסיס 16 | בינארי | ייצוג |
|---|---|---|---|---|
| 105 | 151 | 069 | 01101001 | i |
| 106 | 152 | 06A | 01101010 | j |
| 107 | 153 | 06B | 01101011 | k |
| 108 | 154 | 06C | 01101100 | l |
| 109 | 155 | 06D | 01101101 | m |
| 110 | 156 | 06E | 01101110 | n |
| 111 | 157 | 06F | 01101111 | o |
| 112 | 160 | 070 | 01110000 | p |
| 113 | 161 | 071 | 01110001 | q |
| 114 | 162 | 072 | 01110010 | r |
| 115 | 163 | 073 | 01110011 | s |
| 116 | 164 | 074 | 01110100 | t |
| 117 | 165 | 075 | 01110101 | u |
| 118 | 166 | 076 | 01110110 | v |
| 119 | 167 | 077 | 01110111 | w |
| 120 | 170 | 078 | 01111000 | x |
| 121 | 171 | 079 | 01111001 | y |
| 122 | 172 | 07A | 01111010 | z |
| 123 | 173 | 07B | 01111011 | { |
| 124 | 174 | 07C | 01111100 | \| |
| 125 | 175 | 07D | 01111101 | } |
| 126 | 176 | 07E | 01111110 | ~ |
| 127 | 177 | 07F | 01111111 | DEL |

כפי שניתן לראות, אין בהגדרת ASCII הבסיסית ייצוג לאותיות עבריות. מכיון ש-ASCII ממומש על ידי בית חתום (signed byte) ומאפשר הגדרה על פי 7 סיביות (bit) בלבד (הסיבית הנותרת משמשת להגדרת סימן המספר), ניתן לייצג 127 אותיות וסימנים. כדי להכיל אותיות וסימנים נוספים, יש להשתמש בפורמטים אחרים כגון iso-latin-1 הממומשים על ידי בית לא חתום (unsigned byte) ומאפשרים להגדיר 255 אותיות וסימנים. בדוגמה הבאה נכתוב קוד VBScript (צד לקוח) קצר אשר יציג 255 אותיות וסימנים.

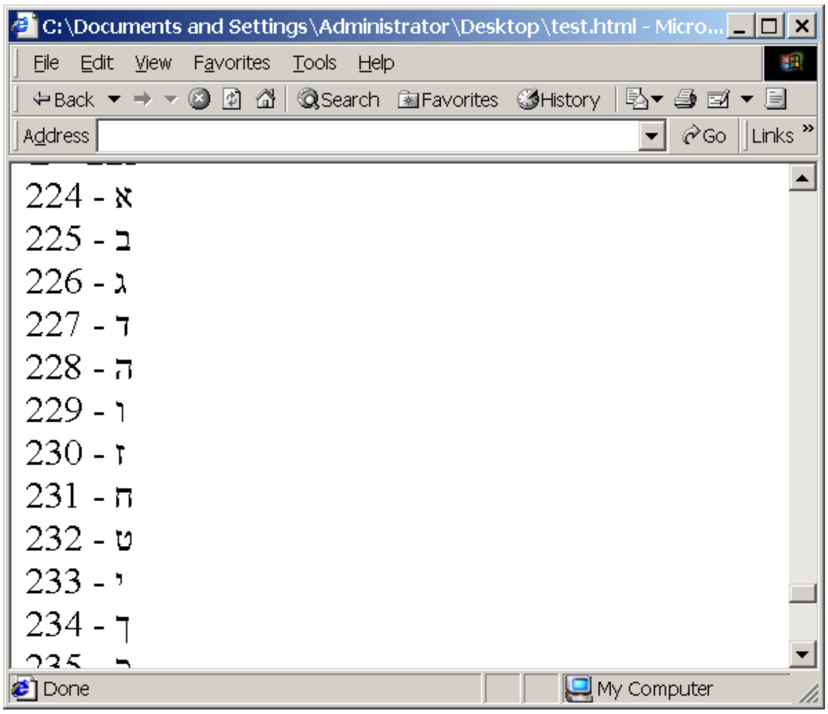
char.html

```
<script language="vbscript">

for i=0 to 255
        document.write chr(i) & " - " & i & "<br>"
next
</script>
```

קוד זה ירוץ 255 פעמים ובכל ריצה יציג את האות או הסימן המיוצגים על ידי i וכן את המספר אותו מכיל i.

בתוצאה בדפדפן, ניתן לראות כי לאותיות העבריות ייצוג החל ממספר 224 עד מספר 250.




למרות שלכאורה נפתרה בעיית העברית על ידי שימוש בפורמטים כגון ISO-LATIN, עדיין נותרה בעיה משמעותית בכתיבת עברית. למרות שבכלים כגון פנקס הרשימות, ניתן לכתוב בעברית (במערכות הפעלה תומכות עברית), נשמרות האותיות על פי סדר הזנתן וכאשר אמור הדפדפן להציגן, נכתבות האותיות בסדר הנכון אך משמאל לימין. כלומר, המילה "שלום", תיכתב כ"םולש".

כאשר נתקלו ראשוני המפתחים ב-Web בבעיה זו, הפך המושג 'הפיכון' למטבע לשון. מלכתחילה המלל בעברית נכתב הפוך. על ידי שימוש בתוכנות קטנות הנקראות 'הפיכון', ההופכות את המלל, כאשר המלל נקרא על ידי הדפדפן הוא מוצג נכון וניתן לקרוא אותו.

שיטה זו ידועה כשיטה הוויזואלית. שיטה זו נתמכת בשני הדפדפנים העיקריים, Netscape ו-Internet Explorer. כל שיש לעשות הוא לכתוב את המלל הפוך ולהגדיר לדפדפן מערך תווים (Character set) המכיל את האותיות העבריות בערכים 224 עד 250. במקרים כאלה, כותב הדפדפן את העברית משמאל לימין ומציג עברית קריאה. כמובן, שמכיון שהשורות מודפסות על המסך מימין לשמאל, נוצרת בעיה חמורה בשבירת שורות.

כלומר, משפט באנגלית : Welcome to Jamaica, יישבר כך :

Welcome
to Jamaica

ואילו משפט בעברית : ברוכים הבאים לג׳מייקה, יישבר כך :

לג׳מייקה
ברוכים הבאים

לכן השיטה הוויזואלית לוקה בחסר ואינה מספקת סביבת עבודה נוחה לפיתוח מערכות בעברית.

הדפדפן Internet Explorer תומך בשיטה הלוגית בה ניתן להקליד עברית באופן רגיל, והדפדפן מטפל בהצגה הנכונה. לא כל הגולשים משתמשים בדפדפן זה, ולכן קיימת בעיה חמורה בהחלטה על שיטת כתיבת העברית. לבעיה זו פתרונות רבים ומגוונים. בספר זה בחרנו להציג פתרון פשוט אשר מיישם את יצירת אובייקט COM להפיכת מלל בעברית עבור דפדפן מסוג Netscape. היתרון בפתרון זה הינו כתיבת המלל פעם אחת באופן רגיל, והפיכה אוטומטית עבור דפדפנים אשר אינם תומכים בעברית לוגית.

ההיגיון מאחורי אובייקט הפיכת מלל הוא כפול. קיים הטיפול במילים ובאותיות, וקיים הטיפול בשורות, מכיון שבעברית ויזואלית קיימת בעיית שבירת השורות. כדי לטפל בשורות יש לזהות בכל ריצה את המספר המייצג שבירת שורה (13) ולהזין כל שורה למערך. הצגת המערך כשורות נפרדות תפתור את בעיית שבירת השורות. כמו כן, הפיכת טקסט אינה עניין פשוט, מכיון שלעיתים קיים מלל המשלב עברית ואנגלית וכן סימנים בעייתיים כגון סוגריים וכד׳.

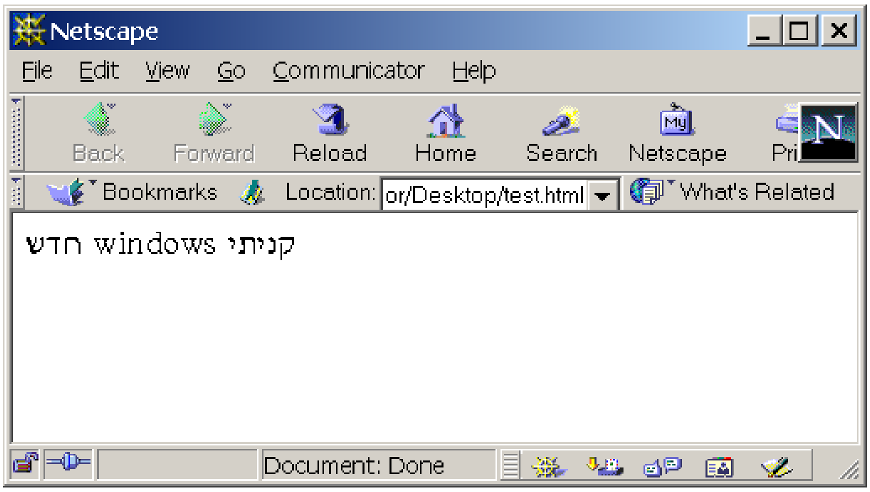
למעשה, מקבל האובייקט מחרוזת ומתחיל לעבור עליה. כל עוד שאין מילים בעברית, כלומר ערכים מספריים שאינם בין 224 ל- 250 (או אשר הינם בין 65 ל- 122), מצרף האובייקט את האותיות למחרוזת המיוצגת על ידי המשתנה newStr. כאשר מזהה האובייקט רווח (ערך מספרי של 32), הוא מניח כי הסתיימה מילה וערכו של המשתנה newStr מוזן למערך המילים הנקרא wordArr.

אותיות עבריות מוזנות לתוך המשתנה tmpStr ועם זיהוי מעבר לשימוש באותיות אנגליות, הופך האובייקט את סדר האותיות העבריות במשתנה tmpStr ומשרשר אותן למשתנה newStr.

כך שבמשפט ״קניתי windows חדש״ יתחיל האובייקט וינתח את המחרוזת משמאל לימין. המילה חדש תזוהה כמילה עם זיהוי הרווח בינה לבין המילה windows ותשורשר הפוך למחרוזת - ״שדח״. המילה ״windows״ אשר מכילה אותיות אנגליות בלבד, תשורשר כמו שהיא אל המחרוזת, כך שנקבל: ״windows שדח״. ולסיום, תשורשר המילה ״קניתי״ במהופך ונקבל את המחרוזת : ״יתינק windows שדח״.

כדי לסיים את התהליך, תוזן כל מילה (שתזוהה על ידי הרווח בין המילים) אל המערך wordArr. התוצאה שיחזיר האובייקט הינה מערך wordArr בסדר הפוך, כך שנקבל את המשפט : ״שדח windows יתינק״.

אם נציג משפט זה בדפדפן Netscape, נקבל את התוצאה:




נתחיל ונבנה את האובייקט ב- Visual basic:

נייצר פרויקט בשם HebTools. בתוכו ניצור את ה-Class: Flip

נפתח את השיטה flipIt אשר תקבל ותחזיר מחרוזת:

Public function flipIt(str as String) as String

נכריז על המשתנים הבאים לפי הסדר הבא,

משתנה עבור הכלת אותיות עבריות וסימנים מיוחדים עד להפיכה עם זיהוי סיום מילה (זיהוי רווח) או זיהוי אות שאיננה בעברית:

Dim tmpStr

משתנה עבור המחרוזת הכוללת (לאחר הפיכת האותיות העבריות):

Dim newStr

משתנה מסוג מערך להכלת כל המילים:

Dim wordArr()

משתנה שיכיל את גודל מערך המילים:

Dim arrSize

משתנה להכלת כל אות בזמן בדיקתה:

Dim currentChr

משתנה שיכיל את האות הקודמת במחרוזת:

Dim preChr

משתנה שיכיל את האות הבאה במחרוזת:

Dim nextChr

נאתחל את משתנה arrSize בערך 1 :

```
Arrsize=1
```

נוסיף למחרוזת שנתקבלה (str) תו רווח בסופה כדי לאפשר בדיקת סיום מילה על ידי איתור רווח בסופה :

```
str=str & " "
```

נתחיל לולאה אשר תרוץ על המחרוזת שנתקבלה ותבדוק אותה מתו מספר 1 לכל אורכה :

```
For i=1 to len(str)
```

נזין את האות הנוכחית למשתנה currentChr, על ידי הפונקציה mid המאפשרת גזירת אות מתוך מחרוזת בתחביר - mid(string to cut, from, number of characters) :

```
CurrentChr=mid(str,clng(i),1)
```

נזין למשתנים שהכנו מראש את האות הקודמת והאות הבאה במחרוזת תוך בדיקה המונעת גלישה מתחומי המחרוזת :

```
If i<len(str)-1 then
	nextChr=mid(str,clng(i+1),1)
Else
	nextChr=" "
End if
If i>1 then
	preChr=mid(str,clng(i-1),1)
Else
	preChr=" "
End if
```

כעת מגיע החלק המסובך באלגוריתם. מכיון שבכל זיהוי של רווח, מוזנות המילים למערך אשר בסיום התהליך מציג את כל המילים בסדר הפוך, יש להימנע ממצב בו שתי מילים באנגלית יוזנו בסדר ישר ויוצגו בסדר הפוך. נבדוק בכל רווח כי המילה הקודמת והבאה אינן אנגליות. אם הרווח נמצא בין שתי מילים באנגלית, נצרף את שתי המילים לאיבר אחד במערך וכך תימנע הפיכתן :

```
If asc(currentChr)=32 and not(asc(nextChr)>64 and asc(nextChr)<123) and_
asc(preChr)>64 and asc(preChr)<123) then
```

אם מתקיים התנאי, נניח כי הסתיימה מילה חדשה. נגדיר שוב את המערך ונזין לתוכו את המילה :

```
Redim preserve wordArr(arrSize)
WordArr(arrSize-1)=newStr & strReverse(tmpStr) & " "
TmpStr=" "
NewStr=" "
ArrSize=arrSize+1
```

אם התנאי אינו מתקיים, נניח כי הרווח הינו בין מילים באנגלית, או שהבדיקה נמצאת באמצע מילה ונמשיך את עיבוד המידע על ידי בדיקת תוכן כל תו. הבדיקה תוודא כי מדובר באות עברית או בתו מיוחד:

```
Else
If (asc(currentChr)>223 and asc(currentChr)<251 or asc(currentChr)=39 _
  or asc(currentChr)=33 _
  or asc(currentChr)=63 _
  or asc(currentChr)=44 _
  or asc(currentChr)=126 _
  or asc(currentChr)=64 _
  or asc(currentChr)=35 _
  or asc(currentChr)=36 _
  or asc(currentChr)=37 _
  or asc(currentChr)=38 _
  or asc(currentChr)=42 _
  or asc(currentChr)=58 _
  or asc(currentChr)=46 _) then
  tmpStr=tmpStr+currentChr
else
  newStr=newStr+strReverse(tmpStr)+currentChr
  tmpStr=" "
end if
end if
```

נתחום את הלולאה:

```
Next
```

כעת נציג את המילים שנאגרו במערך wordArr בסדר הפוך:

```
For z=arrSize-1 to 1 step –1
Flipit=flipit+wordArr(z-1)
Next
```

נסיים את הפונקציה:

```
End function
```

להלן הקוד המלא:

hebToolsDllSourceCode.txt

```
Public Function flipIt(str As String) As String
Dim wordArr() As String
Dim tmpStr
Dim newStr
Dim arrSize As Integer
arrSize = 1
```

```
str = str & " "
For i = 1 To Len(str)
currentChr = Mid(str, CLng(i), 1)
If i < Len(str) - 1 Then
nextChr = Mid(str, CLng(i + 1), 1)
Else
nextChr =" "
End If
If i > 1 Then
preChr = Mid(str, CLng(i - 1), 1)
Else
PreChr=" "
End If
If Asc(currentChr) = 32 And Not_
 (Asc(nextChr) > 64 And _
Asc(nextChr) < 123 And_
 Asc(preChr) > 64 And_
 Asc(preChr) < 123) Then
            ReDim Preserve wordArr(arrSize)
            wordArr(arrSize - 1) = newStr & _
 StrReverse(tmpStr) & " "
            tmpStr = ""
            newStr = ""
            arrSize = arrSize + 1
Else
            If (Asc(currentChr) > 223 And_
 Asc(currentChr) < 251 Or_
 Asc(currentChr) = 39 Or_
 Asc(currentChr) = 33 Or_
 Asc(currentChr) = 63 Or_
 Asc(currentChr) = 44 Or_
 Asc(currentChr) = 126 Or_
 Asc(currentChr) = 64 Or_
 Asc(currentChr) = 35 Or_
 Asc(currentChr) = 36 Or_
 Asc(currentChr) = 37 Or_
 Asc(currentChr) = 38 Or_
 Asc(currentChr) = 42 Or_
 Asc(currentChr) = 58 Or_
 Asc(currentChr) = 46) Then
                        tmpStr = tmpStr & currentChr
            Else
             newStr = newStr & StrReverse(tmpStr) &_
             currentChr
```

```
            tmpStr =" "
          End If
End If
Next
For z = arrSize - 1 To 1 Step -1
flipIt = flipIt & wordArr(z – 1)
Next
End Function
```

כל שנותר הוא לרשום את ה-DLL ולפנות אליו מקובץ ASP אשר יהפוך את המלל במידה ומדובר בדפדפן Netscape. לצורך הדגמה של שימוש באובייקט, נשתמש בקובץ ASP פשוט אשר יאפשר הזנה של מלל לתוך שדה טקסט, והדפסתו לאחר היפוך תוך שימוש בעברית ויזואלית:

Flip.asp

```
<%set flipper=Server.CreateObject("hebTools.flip")
hebStr=flipper.flipIt(request.QueryString("hebTxt"))%>
<form>
<font face="MS Sans Serif"><input name=hebTxt></font>
<input type="submit" value="Flip Now!">
</form>
<br>
<p align=right>
<%=hebStr%>
</p>
```

**הודעה חשובה:**

השימוש בגופן "MS Sans Serif" חיוני להזנת מלל בעברית לשדות טופס ב-Netscape.

חשוב לציין כי טיפול בהפיכת מלל הינו מסובך ואמור להכיל מקרי קצה רבים. המטרה בספר זה אינה לפתור את בעיית הפיכת העברית, אלא להבהיר את השימוש ברכיבי COM.

## פרק 15

# אובייקט ASP Error – ASP 3.0 ומעלה

אובייקט זה אחראי לזיהוי ולכידת אירועי תקלה בזמן ריצת קוד ASP. היישום העיקרי לאובייקט זה הוא איתור תקלות בזמן ריצה לצרכי Debug ומניעת הופעת מסכי Error ג'נריים (כלליים) בזמן אוויר באתרי אינטרנט ואינטראנט.

כדי לייצר אובייקט Error יש להשתמש בתחביר:

```
set e=server.getLastError
```

מרגע שייצרנו את אובייקט Error, ניתן להשתמש בכל אחת מ- 9 התכונות הקיימות שלו.

## ASPCode

תכונה זו מחזירה כמחרוזת את קוד השגיאה שהתרחשה:

```
e.aspcode
```

## Description

תכונה זו מחזירה מחרוזת המכילה תיאור קצר של השגיאה:

```
e.description
```

## ASPDescription

תכונה זו מחזירה תיאור מפורט יותר של השגיאה:

```
e.aspdescription
```

## Category

תכונה זו מחזירה את התחום בו התבצעה השגיאה:

e.Category

## file

תכונה זו מחזירה את שם קובץ ASP בו אירעה השגיאה:

e.file

## Column

תכונה זו מאפשרת לאתר אות מסוים בקובץ ASP בה התבצעה השגיאה:

e.column

## line

בתכונה זו ניתן לאתר את השורה בה התבצעה השגיאה:

e.line

## number

תכונה זו מחזירה את מספר השגיאה במידה והשגיאה התחוללה כתוצאה מפעילות באובייקט COM:

e.number

## source

תכונה זו מחזירה את השורה עצמה בה התבצעה השגיאה:

e.source

# פרק 16

# ASP Reference

## Application

משתנה גלובלי משותף לכל דפי ASP ביישומי WEB.

משתנה מסוג Application שומר על המידע שבו, כל עוד שרת web פעיל.

```
Application("counter") = 1
```

## שיטות

❖ Lock

חוסם את הגישה למשתנה Application כדי למנוע גישה של מספר משתמשים בו-זמנית למשתנה:

```
Application.lock
```

❖ UnLock

משחרר את נעילת משתנה Application לשימוש על ידי משתמשים אחרים:

```
Application.unlock
```

דוגמה:

```
Application.lock  ' locks application to other users
Application("counter")=application("counter") + 1
Application.unlock 'releases the application for other users
```

מומלץ למעט בשימוש בשיטות אלו למניעת מצבים בהם מבקרים באתר לא יוכלו לגשת למשתני Application.

❖ Contents.Remove (ASP 3.0)

מאפשר הסרת תוכן של משתנה אפליקציה על פי שמו, או על פי מיקומו, על ידי ציון ערך מספרי:

```
Application("yossef")="koo koo"
Application("dogli") = "pooki"
```

```
Application.contents.remove(yossef) 'removes the content of
'application("yossef") by name
Or
Application.contents.remove(0) 'removes the content of 'application("yossef") by number
```

❖ Contents.removeAll (ASP 3.0)

מאפשר הסרת כל תכני אובייקט Application:

```
Application.contents.removeAll()
```

# אירועים

❖ OnEnd

אירוע המאפשר כתיבת קוד למצב בו האפליקציה מסתיימת באופן יזום בשרת web. קטע קוד זה נכתב בתוך מסמך Global.asa תחת שגרת קוד בעלת שם קבוע:

```
Sub Application_OnEnd()
' your code goes here
End Sub
```

בתוך שגרת הקוד ניתן לכתוב כל קטע קוד שיטפל במצב זה של סיום אפליקציה.

❖ OnStart

אירוע המאפשר כתיבת קוד למצב בו האפליקציה מתחילה (עם כניסת המשתמש הראשון לדף ASP). קטע קוד זה נכתב בתוך מסמך Global.asa תחת שגרת קוד בעלת שם קבוע:

```
Sub Application_OnStart()
' your code goes here
End Sub
```

בתוך שגרת הקוד ניתן לכתוב כל קטע קוד שיטפל במצב זה של התחלת אפליקציה.

# מאפיינים

❖ Contents (nameOfApplication)

מכיל את כל האובייקטים מסוג Application ומאפשר פנייה אליהם על ידי ציון שמם (nameOfApplication):

```
Application("banana") = "split"
Application("shish") = "kabab"
Application("mooki") = "pooki"
 For Each nameOfApplication in Application.Contents
    Response.Write  Application.Contents(nameOfApplication)
End If
Next
```

# Request

אובייקט זה מכיל בתוכו את כל המידע שנשלח מהדפדפן אל השרת.

## מאפיינים

❖ Cookies (nameOfCookie)

מאפשר קריאה של העוגיות שנמצאות אצל הלקוח לפי שמם.

Request.Cookies("ugiya") ' gets the value of the cookie named "ugiya"

❖ Form(fieldName)

מאפשר פנייה אל השדות השונים שנשלחו בטופס אל השרת על פי שמם, בעזרת שיטת Post.

Request.Form ("Age") ' gets the value of the field "age"

❖ QueryString (fieldName)

מאפשר פנייה אל השדות השונים שנשלחו אל השרת על פי שמם בעזרת שיטת Get.

Request.QueryString ("lastName") ' gets the value of "lastName"

❖ ServerVariables (variableName)

מאפשר גישה למשתנים קבועים מראש בצד השרת.

| | |
|---|---|
| **ALL_HTTP** | כותרת HTTP שנשלחה על ידי הלקוח |
| **ALL_RAW** | מחזיר את השדות בכותרת בצורתם הגולמית |
| **APPL_PHYSICAL_PATH** | מחזיר את הנתיב בו מאוחסנים קבצי האפליקציה |
| **AUTH_PASSWORD** | הסיסמה שהקליד המשתמש בתיבת הדו-שיח בדפים מאובטחים (Basic authentication) |
| **AUTH_TYPE** | באיזו שיטה משתמש השרת לאבטחת קבצים |
| **AUTH_USER** | שם המשתמש מתוך תיבת הדו-שיח בדפים מאובטחים |
| **CONTENT_LENGTH** | אורך המידע שנשלח מהלקוח |
| **CONTENT_TYPE** | סוג המידע שנשלח מהלקוח |
| **GATEWAY_INTERFACE** | גרסת CGI בשרת |
| **HTTP_<*HeaderName*>** | פנייה לכותרת HTTP על פי שמה |

| | |
|---|---|
| **HTTPS** | מחזיר NO תחת SSl או OFF תחת ערוץ שאינו מאובטח |
| **LOCAL_ADDR** | מחזיר את כתובת השרת ממנו הגיעה הבקשה |
| **PATH_INFO** | הנתיב הווירטואלי של הקבצים בשרת |
| **PATH_TRANSLATED** | הנתיב הפיסי של הקבצים בשרת |
| **QUERY_STRING** | מחזיר את המידע שנשלח באמצעות GET |
| **REMOTE_ADDR** | כתובת IP של הלקוח |
| **REMOTE_HOST** | שם המחשב של הלקוח |
| **REMOTE_USER** | שם משתמש לא ממופה כפי שנשלח על ידי הלקוח |
| **REQUEST_METHOD** | השיטה בה נעשה שימוש לצורך Request |
| **SCRIPT_NAME** | הנתיב הווירטואלי של הקובץ המבוקש על ידי הלקוח |
| **SERVER_NAME** | כתובת השרת |
| **SERVER_PORT** | כתובת Port אליו נשלח Request |
| **SERVER_PORT_SECURE** | מחזיר 1 במידה ו-Request מטופל על ידי SSL, או 0 במידה ולא |
| **SERVER_PROTOCOL** | באיזו גרסת פרוטוקול מטפל Request |
| **SERVER_SOFTWARE** | גרסת ושם השרת המטפל בפנייה |

```
Request.serverVariables("REMOTE_ADDR")
```

# Response

## שיטות

❖ AddHeader headerName,HeaderValue

הוספת כותרת HTTP על ידי ציון שמה וערכה:

```
Response.AddHeader "newHeader", "Hello"
```

❖ AppendToLog (stringToAdd)

הוספת שורה לסוף קובץ Log בשרת עבור הבקשה הנוכחית על ידי ציון המחרוזת להוספה. אורך המחרוזת המקסימלית שניתן להוסיף הוא עד 80 תווים ואינו יכול להכיל פסיקים (,):

```
Response.AppendToLog("this is a great book")
```

❖ Clear

מנקה את החוצץ (Buffer). אם לא קיים חוצץ תתקבל הודעת שגיאה:

```
Response.Clear
```

❖ End

מסיים את עיבוד התסריט (Script) ושולח את הקובץ ללקוח ללא עיבוד נוסף:

```
Response.Write "This will get to the client"
Response.End
Response.Write "this will not!!!"
```

❖ Flush

שולח את תוכן החוצץ ללקוח (Buffer):

```
Response.Flush
```

❖ Redirect  WhereToGo

שולח דף מסוים חזרה ללקוח. פעולה זו יכולה להתבצע רק לפני שנשלחו ללקוח חזרה פקודות HTML:

```
Response.Redirect "startPage.asp"
```

❖ Write whatToWrite

שולח חזרה לדפדפן HTML:

```
Response.write "Hello Shooki!"
```

## מאפיינים

❖ Buffer

מורה לשרת לסיים את עיבוד הדף ורק בסיומו לשלוח חזרה את התוצאה ללקוח. ההכרזה על מאפיין זה חייבת להיעשות בתחילת הדף:

```
Response.Buffer = TRUE
```

❖ CacheControl publicOrPrivet

הכרזה על מאפיין זה מאפשרת לשרתי proxy לשמור את הדף ב-Cache.

Public – מאפשר אחסון

Privet – אינו מאפשר אחסון

```
Response.CacheControl = "Public"
```

❖ Charset (charsetToBeUsed)

ערכת התווים בה ישתמש הדפדפן לפענוח תווי הטקסט:

```
Rsponse.Charet  ("iso-8859-8")
```

❖ Response.ContentType

מורה לדפדפן באיזו אפליקציה להשתמש כדי להציג את הדף:

```
Response.ContentType = "application/vnd.ms-excel"
```

❖ Expires

מורה לדפדפן מתי לטעון את הדף חזרה מהשרת ולא מה-Cache. הערך נקבע בדקות:

```
Response.Expires = 10
```

❖ ExpiresAbsolute

קובע מתי הדפדפן ינקה את הדף מה- Cache על ידי ציון תאריך מדויק:

את השנה יש לציין באמצעות ארבע ספרות. את השעה יש לציין בפורמט של 24 שעות:

```
Response.ExpiresAbsolute=#March 21, 2000 18:00:00#
```

❖ IsClientConnected

מעדכן את השרת אם הלקוח עדיין מחובר אליו מאז Response.Write האחרון. תשובה חיובית תחזיר True. תשובה שלילית תחזיר False:

```
If response.isclientconnected=false then
...
End if
```

# Collection

❖ Cookies (nameOfCookie) = value

מאפשר שליחת עוגיות ללקוח. יש לשלוח עוגיות לפני שליחת HTML ללקוח ולהוסיף את הפרמטר Expires (ציון תאריך תפוגה), ואת הפרמטר Path לקביעת הנתיב ממנו ניתן יהיה לקרוא את העוגיה:

```
Response.Cookies("melafefon") = "2 kilo"
Response.cookies ("melafefon").expires = "july 21,2002"
Response.Cookies("melafefon").path= "/"
```

# Server

## שיטות

❖ CreateObject (whatObject)

יצירת מופע חדש של אובייקט מהשרת. יש להשתמש במילה Set לפני הגדרת שם המופע החדש:

```
Set newObject = Server.CreateObject("objectBank.coolObject")
```

❖ HTMLEnode (stringToDisplay)

מאפשר כתיבת תגי HTML ללקוח מבלי שהדפדפן יבצע אותם:

```
Response.Write Server.HTMLEncode("the body tag is: <body>")
```

❖ URLEnode (stringToDisplay)

מאפשר כתיבת מחרוזת על פי פורמט URL תקני לוודאות תקינות פורמט הקישור:

```
Response.Write Server.URLEncode("http://www.mightyJoeYoung.com")
```

## מאפיינים

❖ ScriptTimeout

קובע את משך הזמן המוקצב לביצוע תסריט (Script). משך הזמן נקבע בשניות:

```
Server.ScriptTimeout = 135
```

# Session

אובייקט זה מאותחל אוטומטית עבור כל משתמש בדף ASP הראשון אותו הוא טוען. האובייקט קיים עבור כל משתמש כל עוד הדפדפן פתוח ולא עבר יותר מהזמן המוגדר מראש (ברירת מחדל 20 דקות).

## שיטות

❖ Abandon

מסיים את אובייקט Session עבור לקוח מסוים:

```
Session.Abandon
```

## אירועים

❖ OnEnd

אירוע המאפשר כתיבת קוד למצב בו Session מסתיים, בין אם באופן יזום או באמצעות סגירת הדפדפן או פקיעת הזמן המוקצב. קטע קוד זה נכתב בתוך מסמך Global.asa תחת שגרת קוד בעלת שם קבוע:

```
Sub Session_OnEnd()
' your code goes here
End Sub
```

בתוך שגרת הקוד ניתן לכתוב כל קטע קוד שיטפל במצב זה של סיום Session.

❖ OnStart

אירוע המאפשר כתיבת קוד למצב בו Session מתחיל (עם כניסת המשתמש לדף ASP הראשון). קטע קוד זה נכתב בתוך מסמך Global.asa תחת שגרת קוד בעלת שם קבוע:

```
Sub Session_OnStart()
' your code goes here
End Sub
```

בתוך שגרת הקוד ניתן לכתוב כל קטע קוד שיטפל במצב זה של התחלת Session.

## מאפיינים

❖ LCID

קובע את המיקום הגיאוגרפי של השרת כדי לתת נתוני תאריך ושעה מדויקים לאותו אזור זמן:

```
Session. LCID = 1041 ' Japan time zone
```

❖ SessionID

מחזיר את מאפיין ID של Session שהינו מאפיין ייחודי עבור כל משתמש והינו מסוג Long:

```
Session.SessionID
```

❖ TimeOut

קובע מתי יש להפסיק את אובייקט Session עבור לקוח, במידה ועובר פרק זמן מסוים ללא תקשורת בין השרת ללקוח. הקביעה נעשית בדקות (ברירת המחדל היא 20 דקות):

```
Session.timeout = 30
```

❖ Contents (nameOfSession)

מכיל את כל האובייקטים מסוג Session ומאפשר פנייה אליהם על ידי ציון שמם (nameOfSession):

```
Session("mar") = "olam"
Session("bar") = "baaba"
Session("doda") = "tzipora"


For Each nameOfSession in Session.Contents
    Response.Write  Session.Contents(nameOfSession)
End If
Next
```

## פרק 17

# VBScript Reference - Server Side

## Class object

מינוח Class מתייחס לאובייקטים שנוצרו באמצעות vbscript בעזרת הכרזת :

```
Class…
…
End Class
```

ויצירת מופע חדש שלהם על ידי שימוש ב-Set ו-New :

```
Dim ObjectVariable
Set ObjectVariable = New className
```

## Dictionary object

מאפשר יצירת אובייקט המכיל צמדים של מפתח וערך ובכך מאפשר פנייה לערכים השונים באובייקט Dictionary :

```
Dim foods
Set foods = CreateObject("Scripting.Dictionary")
foods.Add "a", "Bannana"
foods.Add "b", "humus"
foods.Add "c", "jahcnun"
Response.Write "foods.Item("b")"
```

## שיטות

### Add

הוספת צמד חדש לאובייקט בפורמט מפתח/ערך:

```
myDictionaryObject.add key,value
```

### exists

שיטה לבדיקה האם ערך מסוים קיים באובייקט או לא. השיטה מחזירה True במידה והתשובה חיובית ו-False במידה ותשובה שלילית:

```
myDictionaryObject.exists(key)
```

### Items

מחזיק את כל תכולת אובייקט Dictionary בתוך מערך:

```
NewArray = dictionaryObjectName.items
```

### Keys

מחזיק את כל תכולת מפתחות Dictionary בתוך מערך:

```
NewArray = dictionaryObjectName.keys
```

### Remove

מאפשר הסרת צמד ערך/מפתח מתוך אובייקט Dictionary על ידי ציון מפתח מסוים. במידה ומפתח זה אינו קיים, תתקבל הודעת שגיאה:

```
DictionaryObject.remove("b")
```

מאפשר הסרת כל צמדי ערך/מפתח מתוך אובייקט Dictionary:

```
DictionaryObject.removeall
```

### Count

מחזיר את מספר זוגות מפתח/ערך בתוך אובייקט Dictionary:

```
DictionaryObject.count
```

### Item

מאפשר שינוי/הוספת ערך לאובייקט Dictionary:

```
DictionaryObject.item("a") = "Babushka"
```

## Key

מאפשר שינוי שיוך מפתח לערך :

```
DictionaryObject.key(oldKey) = newKey
```

# Drive Object

אובייקט זה מאפשר גישה למידע על כוננים שונים, מקומיים או ברשת :

```
Set fs = CreateObject("Scripting.FileSystemObject")
Set drv = filesys.GetDrive("c")
```

## מאפיינים

### AvailableSpace

מחזיר את המקום הפנוי בכונן מסוים :

```
driveObject.AvailableSpace
```

### DriveLetter

מחזיר את אות הכונן אליו אנו פונים :

```
object.DriveLetter
```

### DriveType

מחזיר מספר המציין את סוג הכונן לפי הערכים הבאים :

- 0- לא ידוע
- 1- ניתן להסרה
- 2- קבוע
- 3- רשת
- 4- תקליטורים
- 5- RAM

```
driveObject.DriveType
```

### FileSystem

מחזיר את סוג מערכת ניהול הקבצים בכונן מסוים :

```
driveObject.FileSystem
```

## FreeSpace

מחזיר את כמות המקום הפנוי בכונן מסוים:

```
driveObject.FreeSpace
```

## IsReady

מחזיר ערך True אם הכונן זמין לשימוש, מחזיר False אם לא:

```
driveObject.IsReady
```

## Path

מחזיר את הנתיב עבור קובץ, תיקיה או כונן:

```
driveObject.Path
```

## SerialNumber

מחזיר ערך עשרוני של המספר הסדרתי של הכונן:

```
driveObject.SerialNumber
```

## ShareName

מחזיר שם שיתוף של כונן מסוים ברשת על פי תקן UNC, לשימוש בעת עבודה מול כונן מרוחק בלבד:

```
driveObject.ShareName
```

## TotalSize

מחזיר נפח כונן מסוים בבתים:

```
driveObject.TotalSize
```

## VolumeName

קובע או מחזיר שם כונן מסוים.

# Err object

אובייקט זה מכיל את מספר השגיאה האחרונה שהתרחשה בזמן ריצה. אין צורך לייצר מופע של אובייקט זה כיון שהוא נוצר אוטומטית.

## מאפיינים

### Description

מכיל כברירת מחדל תיאור קצר של השגיאה שהתחוללה בזמן הריצה. ניתן להזין תיאור אחר אם מתעורר הצורך:

```
Err.description = "Oops…I just formatted your Hard Drive!"
Response.write Err.description
```

### Number

מאפיין ברירת המחדל של אובייקט Err. מחזיר את מספר השגיאה שהתרחשה:

```
Err.number
```

# FileSystemObject object

אובייקט המאפשר גישה לקבצים לצורך יצירת הוספה או שינוי. יש לייצר מופע של אובייקט זה.

```
Set fileObject = CreateObject("Scripting.FileSystemObject")
```

## שיטות

### CopyFile

שיטה המאפשרת העתקת קבצים לתיקיה חדשה. ניתן להוסיף פרמטר אופציונלי בעל ערכים של True ו-False, לקביעה האם לדרוס קובץ קיים בתיקיית היעד בעל שם דומה:

```
fileObject.CopyFile sourceFile, destinationFolder [, overwrite]
```

### CopyFolder

שיטה המאפשרת העתקת קבצים לתיקיה חדשה. ניתן להוסיף פרמטר אופציונלי בעל ערכים של True ו-False, לקביעה האם לדרוס קובץ קיים בתיקיית היעד בעל שם דומה:

```
fileObject.CopyFolder sourceFolder, destinationFolder [, overwrite]
```

## CreateFolder

שיטה המאפשרת יצירת תיקיה:

```
fileObject.CreateFolder foldername
```

## CreateTextFile

שיטה המאפשרת יצירת קובץ טקסט. ניתן להוסיף פרמטר אופציונלי בעל ערכים של True ו-False, לקביעה האם לדרוס קובץ קיים בתיקיית היעד בעל שם דומה:

```
fileObject.CreateTextFile filename [, overwrite[, unicode]]
```

## DeleteFile

מאפשר מחיקת קובץ. מכיל אופציה של True הקובעת מחיקה גם של קבצי קריאה בלבד:

```
fileObject.DeleteFile file [, forceDelete]
```

## DeleteFolder

מאפשר מחיקת תיקיה, מכיל אופציה של True הקובעת מחיקה גם של קבצי קריאה בלבד:

```
fileObject.DeleteFolder folder [, force]
```

## DriveExists

שיטה המחזירה True אם כונן מסוים קיים, ומחזירה False אם לא:

```
fileObject.DriveExists(driveLetter)
```

## FileExists

שיטה המחזירה True אם קובץ מסוים קיים, ומחזירה False אם לא:

```
fileObject.FileExists(fileName)
```

## FolderExists

שיטה המחזירה True אם תיקיה מסוימת קיימת, ומחזירה False אם לא:

```
fileObject.FolderExists(folderName)
```

## GetFile

מחזיר אובייקט File עבור קובץ מסוים:

```
fileobject.GetFile(filename)
```

## GetFolder

מחזיר אובייקט Folder עבור תיקיה מסוימת :

```
fileObject.GetFolder(folderName)
```

## MoveFile

הזזת קבצים ממקום למקום :

```
fileObject.MoveFile sourceFile, destinationFolder
```

## MoveFolder

הזזת תיקיות ממקום למקום :

```
fileObject.MoveFolder sourceFolder, destination
```

## OpenTextFile

מחזיר מופע של אובייקט TextStreamObject לאחר פתיחת קובץ טקסט מסוים.

פרמטרים אופציונליים :

Iomode –8 (ForAppending) להוספה/שינוי
1 (Forreading) לקריאה בלבד
2 (Forwriting) לכתיבה (דריסת הקיים)
Create- true
False
Format- 0 (ASCII)
-1 (Unicode)
-2 (System default)

```
fileObject.OpenTextFile(filename [, iomode[, create[, format]]])
```

## CreateFolder

שיטה המאפשרת יצירת תיקיה:

fileObject.CreateFolder foldername

## CreateTextFile

שיטה המאפשרת יצירת קובץ טקסט. ניתן להוסיף פרמטר אופציונלי בעל ערכים של True ו-False, לקביעה האם לדרוס קובץ קיים בתיקיית היעד בעל שם דומה:

fileObject.CreateTextFile filename [, overwrite[, unicode]]

## DeleteFile

מאפשר מחיקת קובץ. מכיל אופציה של True הקובעת מחיקה גם של קבצי קריאה בלבד:

fileObject.DeleteFile file [, forceDelete]

## DeleteFolder

מאפשר מחיקת תיקיה, מכיל אופציה של True הקובעת מחיקה גם של קבצי קריאה בלבד:

fileObject.DeleteFolder folder [, force]

## DriveExists

שיטה המחזירה True אם כונן מסוים קיים, ומחזירה False אם לא:

fileObject.DriveExists(driveLetter)

## FileExists

שיטה המחזירה True אם קובץ מסוים קיים, ומחזירה False אם לא:

fileObject.FileExists(fileName)

## FolderExists

שיטה המחזירה True אם תיקיה מסוימת קיימת, ומחזירה False אם לא:

fileObject.FolderExists(folderName)

## GetFile

מחזיר אובייקט File עבור קובץ מסוים:

fileObject.GetFile(fileName)

## פרק 18

# SQL Reference

## פקודת Select

פקודה המשמשת לשליפת נתונים מתוך מסד נתונים טבלאי. יש לציין איזה עמודות ברצוננו לשלוף:

```
Select columnName,anotherColumnName,etc..
```

לביטול הצגה כפולה של שמות החוזרים על עצמם יש להשתמש ב– Distinct:

```
select fname,lname, distinct city
```

התוצאה לא תכיל כפילויות של שדה City.

לבחירת כל העמודות מתוך הטבלה יש להשתמש בסימן הכוכבית:

```
Select *
```

## From

שם הטבלה ממנה אנו שולפים את הנתונים. פרמטר חובה:

```
Select * from tableName
```

## Where

הוספת תנאי לשליפת מידע על ידי אופרטורים השוואתיים מופרדים על ידי AND, OR ו-NOT:

| | |
|---|---|
| שוויון | = |
| אי שוויון | <> |
| גדול מ- | > |
| קטן מ- | < |

| | |
|---|---|
| גדול שווה | >= |
| קטן שווה | <= |
| דומה ל- | like |
| אינו דומה | Not like |
| בין לבין | Between...and... |

```
select * from tableName where city='petachTikva' and  size>12
```

## order by

סדר הצגת התוצאות על פי שם טור:

```
select * from tableName order by city
```

לשליפת נתונים מכמה טבלאות בבת אחת יש לציין את שם הטבלה והעמודה אותה אנו רוצים להציג:

```
select tableName1.columnName,tableName2.columnName from tableName1,tableName2
```

## משפט insert

משמש לצורך הזנת נתונים לטבלה.

## into

מציין את שם הטבלה אליה יוזנו הנתונים:

```
insert into TableName
```

## Values

ציון הערכים אותם יש להזין. ערכים מחרוזתיים יש להקיף בגרש בודד מכל צד:

```
insert into tableName values('banana',34,'kookoo','dalit')
```

בצורת הזנה זו ייכנסו הנתונים לטבלה על פי סדר הכתיבה שלהם משמאל לימין. לכן חשוב להקפיד שסדר הערכים זהה לסדר הטורים בטבלה. להזנת חלק מהטורים בלבד יש לציין לאיזה טורים במדויק נזין את הערכים החדשים על ידי הוספת פרמטרים לביטוי Into:

```
insert into(city,fruit) values('raanana','bananna')
```

# משפט Update

משפט המשמש לעדכון נתונים בטבלה. יש לציין באיזו טבלה מדובר:

```
Update tableName
```

## Set

קביעת הערכים החדשים לכל שדה שעובר עדכון:

```
update tableName set city='dimona',fruit='paamona'
```

## Where

באיזו רשימה ספציפית יש להזין את הנתונים המעודכנים:

```
update tableName set city='dimona',fruit='paamona' where fname='yossef'
```

# משפט Delete

## Delete

פקודה למחיקת נתונים מתוך הטבלה.

## from

מאיזו טבלה יש למחוק את הנתונים:

```
delete from TableName
```

## Where

באיזו רשימה ספציפית יש לבצע את מחיקת הנתונים:

```
delete from TableName where name='kookoorikoo'
```

## משפט Create Table

יצירת טבלה חדשה על ידי מתן שם לטבלה, שמות לטורים וסוג המידע בכל טור:

```
create table tableName(columnName dataType,columnName dataType...)
```

## משפט Drop table

משמש למחיקת כל הטבלה על ידי ציון שם הטבלה:

```
drop table TableName
```

# אינדקס

האינדקס הינו באנגלית בלבד. כיוון הקריאה משמאל, מסוף הספר.

כלומר, פתח את הספר בסופו (לפני הקטלוג) ושם תמצא את תחילת האינדקס - עמוד 1.

# INDEX

## B

## C

## D



## E

**F**

## G



## H

**I**



**J**



**L**



**M**



**N**



**O**

## P



## Q



## R

## S

## W